Lie Group Methods in Geometric Integration

Pui-Sze Priscilla Tse

A thesis submitted in partial fulfilment of the requirements for the degree of
Master of Science in Applied Mathematics,
The University of Auckland, 2003.

Abstract

The subject of geometric integration concerns the construction of numerical integrators which can successfully preserve the asymptotic behaviour or invariants of the approximate solutions, computed for certain dynamical systems. Whereas one of the aims when building a classical numerical integrator is to minimise the error produced by the method, the emphasis in the construction of a geometric integrator lies in its ability to produce numerical solutions with the correct qualitative behaviour from the differential equations. This leads to the specific design of numerical integrators to solve certain types of dynamical systems.

It turns out that the language of Lie groups and Lie algebras is particularly suited to the building of geometric integrators, based on their actions on manifolds. A geometric integrator which produces numerical solutions lying on the manifold of a dynamical system via the Lie group action, is an integrator which belongs to a class of numerical methods known as Lie group methods.

The theory behind Lie group methods has been studied extensively by researchers in the University of Bergen and the University of Trondheim, Norway. While the theoretical aspects of Lie group integrators are developing rapidly, the procedures for applying such integrators to solving general problems in physical sciences are not yet fully understood. In this thesis, we shall look at some of the basic concepts behind the two main classes of Lie group methods, namely the Magnus series expansion and the Runge-Kutta-Munthe-Kaas methods. In particular, such Lie group methods are applied to solve several systems of differential equations. The performance of the Lie group integrators are then compared with the standard numerical methods such as the classical Runge-Kutta methods. Preliminary results indicate that Lie group methods are able to preserve some properties of the numerical solutions inherent in the particular dynamical system, but at great computational cost.

Acknowledgements

First and foremost, I would like to thank Prof. John Butcher, for his patience and kindness. His great enthusiasm for mathematics is an inspiration to me, and this thesis would not have been possible without his guidance. I did not have the chance to visit all of the 'destinations' on his 'lonely planet guide to geometric integrations', but under his supervision, I managed to stop by some of the 'famous attractions'!

Next, I wish to thank the members of the Numerical Analysis Group at The University of Auckland, including Dr. Robert Chan, Dr. Allison Heard, Will Wright, Nicolette Moir, Shirley Huang, Angela Tsai, Dr. Pierre Leone and Steffen Schulz. Their support is essential to me, especially in my attempts to explain a subject which I am only slowly beginning to understand, and for having read critically parts of this thesis.

Also, my gratitude goes to Prof. Brynjulf Owren and Dr. Elena Celledoni of The Norwegian University of Science and Technology, and Dr. Tina Chan of The National Taichung Institute of Technology, Taiwan. Their encouragement and willingness in answering my queries over such long distances, is deeply appreciated.

Dr. Vivien Kirk, from the Department of Mathematics at The University of Auckland, is very active in her support for all of her students. I would like to take this opportunity to acknowledge her work as the organiser of the Summer Scholarships Programme. This program provides financial assistances to students over the summer period, and it enabled me to finish my thesis without financial worries.

My sincere thanks go to the 2002 post-graduate mathematics students at The University of Auckland, especially Brian van Dam, Mark Liu, Amy Peny, Sophia Chen, Richard Vale, David Simpson, David Welch, and David Grant. I have come to appreciate the importance of regular contacts with fellow mathematics students. This year would not have been possible for me, without their companionship, laughter and support.

Similarly, I am indebted to the 2002 demonstrators in ADVLAB from the Departments of Mathematics and Statistics at The University of Auckland, including Paul Chang, Brad Luen, Derek Law, Kristy Su, Peter Hu, Janet Pearson, and Wei Han.

Being the only 'non-statistics' demonstrator in ADVLAB was both fun and challenging, especially when the lab was full of statistics students! Thanks for making the lab such a fun place to work in.

Last but not least, a humble acknowledgement goes to the past and present staff at the New Lynn branch of the Waitakere City Library, especially the team coordinator Margaret Cundall, my Saturday supervisors Mohammed Faruk, Robyn Whitwell, and Christine Marter, my co-workers Rhiannon Kovacevich and Pragati Vasisht. Their understanding and encouragement over the past eight years have been tremendous, and their dedication to serving the wider community is a great source of inspiration to me. It will always remind me that not only can personal fulfilment be found in the learning and discovery, but also in the sharing of one's knowledge with other people. The willingness to make a gift of knowledge to others is truly priceless.

Dedication

献给我的家人，感谢他们的爱。

Notations

- $\mathcal{M}$, $\mathcal{N}$ Manifolds.
- G Lie group.
- $\mathfrak{g}$ Associated Lie algebra.
- $\mathfrak{G}$ Universal enveloping algebra of a Lie algebra $\mathfrak{g}$.
- $\mathcal{F}(\mathcal{M})$ The set of all real valued functions.
- $T_p\mathcal{M}$ Tangent space at point $p \in \mathcal{M}$.
- $\mathfrak{X}(\mathcal{M})$ Set of all vector fields on $\mathcal{M}$.
- $\mathbf{d}g$ The differential of function $g \in \mathcal{F}(\mathcal{M})$.
- $F[g]$ The Lie derivative of $g \in \mathcal{F}(\mathcal{M})$ with respect to the vector field $F \in \mathfrak{X}$.
- $\Psi_{F,t}$ The flow of a vector field F.
- Ψ The coordinate mapping from $\mathfrak{g}$ to G.
- $\mathbb{R}$ The set of real numbers.
- $\mathbb{Z}$ The set of integers.
- Id The identity matrix.
- $\mathbb{I}$ Zeroth order identity differential operator in enveloping algebra $\mathfrak{G}$.
- B_s Bernoulli numbers ($B_0 = 1, B_1 = -1/2, B_2 = 1/6,$ and all odd $B_{k\geq 3}$ are 0).
- $\mathbb{T}_k$ Set of terms in the Magnus series expansion containing kth integrals.
- $\mathbb{F}_m$ Set of terms in the Magnus series expansion of power m.
- SO(n) The n-dimensional special orthogonal Lie group.
- $\mathfrak{so}(n)$ The n-dimensional special orthogonal Lie algebra.
- Ad The Adjoint representation.
- ad The adjoint operator.
- exp The exponential map.
- expm Matlab's matrix exponential function.
- dexp Differential of the exponential map.
- cay The Cayley map.
- dcay Differential of the Cayley map.
- ccsk Canonical coordinates of the second kind.
- dccsk The differential of the canonical coordinates of the second kind.
- $\twoheadrightarrow$ The matrix-to-vector representation for skew-symmetric matrices in $\mathfrak{so}(n)$.
- $\twoheadleftarrow$ The vector-to-matrix representation for skew-symmetric matrices in $\mathfrak{so}(n)$.

Contents

List of Tables

List of Figures

CHAPTER 1

Introduction and Preliminary Mathematics

In this chapter we will look at some of the motivation for studying Geometric Integrators, and introduce some of the terminologies which we will be using later on.

1.1 Introduction

1.1.1 Geometric Integration and Motivation for Lie Group Methods

The subject of geometric integration concerns the construction of numerical integrators which can successfully preserve the asymptotic behaviour or invariants of the approximate solutions, computed for certain dynamical systems. Whereas the aim of a classical numerical integrator is to minimise the error produced by the method, the emphasis in the construction of a geometric integrator lies in its ability to produce numerical solutions with correct qualitative behaviour from the differential equations. In Chapter 1 of [14], the inaccuracies of solutions when solving examples like the simple mathematical pendulum or the differential equations describing the outer solar system using classical integrators, serve as a motivation for using geometric concepts to improve the qualitative behaviour of numerical solutions. It becomes apparent that when solving some dynamical systems, the classical methods are inappropriate because, in some cases, it is not enough to reduce the errors produced by the integrator. The integrator should also be designed to produce numerical solutions which preserve certain invariants or the qualitative behaviour of the system.

It turns out that the language of Lie Groups and Lie Algebras is suited for the building of geometric integrators, based on their actions on manifolds. A geometric integrator which produces numerical solutions lying on the manifold of a dynamical system, is an integrator which belongs to a class of numerical methods known as Lie group methods. The theory behind this particular class of methods is a fascinating combination of pure and applied mathematics, and in this thesis, we will explore some of the details and applications of these Lie group methods.

1.2 Preliminary Mathematics

The following are some of the terms which will be mentioned in subsequent chapters. Further explanations and proofs can be found from the list of texts in the Bibliography, especially [1],

[19], [14], and [28].

1.2.1 Differential Manifold $\mathcal{M}$

Before we can discuss the definition of a differentiable manifold, we need to understand the notions of a topological space. Generally speaking, a topological space is a set for which its topology has been specified. From [1], we have the following definition.

Definition 1.1. *A topological space is a set S together with a collection of open sets O of subsets such that,*

1. $\{\emptyset\} \in O$ *and* $S \in O$.
2. *If* $U_1, U_2 \in O$, *then* $U_1 \cap U_2 \in O$.
3. *The union of any collection of open sets is open.*

From the above, the concept of a differentiable manifold $\mathcal{M}$ is defined in [23] and [9].

Definition 1.2. *Let $\mathcal{M}$ be a topological space and an atlas on $\mathcal{M}$ is a collection of charts made up of the pairs (U_i, ϕ_i). Then $\mathcal{M}$ is an n-dimensional differentiable (or smooth) manifold if each of the following hold,*

1. *Each domain U_i of the chart (U_i, ϕ_i) is an open subset of $\mathcal{M}$, and the domains $\{U_i\}$ cover $\mathcal{M}$. That is $\mathcal{M} = \bigcup_i U_i$, where each element in $\mathcal{M}$ belongs to at least one open subset U_i.*
2. *For every chart (U_i, ϕ_i), there exists open subsets $V_i \subset \mathbb{R}^n$ such that ϕ_i is a homeomorphism,*
$$\phi_i : U_i \to V_i \subset \mathbb{R}^n,$$
of U_i onto $V_i \subset \mathbb{R}^n$. Note that homeomorphism is a bijection ϕ_i where ϕ_i and ϕ_i^{-1} are both continuous, and the image and inverse image of open sets are themselves open [9].
3. *For any two charts (U_i, ϕ_i) and (U_j, ϕ_j) such that the intersection $U_i \cap U_j$ is nonempty, the map,*
$$\phi_{ij} = \phi_j \circ \phi_i^{-1} : \phi_i(U_i \cap U_j) \to \phi_j(U_i \cap U_j),$$
is infinitely differentiable for all pairs of i, j.

All of the overlapping charts $\phi_{ij} : V_i \to V_j$ are diffeomorphism (i.e. smooth function with a smooth inverse), as presented in Figure (1.1). Furthermore, a differentiable manifold $\mathcal{M}$ is said to be a submanifold of a differentiable manifold $\mathcal{N}$ if $\mathcal{M} \subset \mathcal{N}$, and the identity map of $\mathcal{M}$ into $\mathcal{N}$ is an embedding (i.e. The properties of $\mathcal{N}$ restricted to $\mathcal{M}$ are the same as the properties of $\mathcal{M}$.) For our purposes, strictly speaking the manifold $\mathcal{M}$ is really a submanifold of dimension d, embedded in the manifold $\mathbb{R}^n$, as stated in [19].

> *A d-dimensional manifold $\mathcal{M}$ is a d-dimensional smooth surface $\mathcal{M} \subset \mathbb{R}^n$ for some $d \leq n$.*

Alternatively, based on local coordinates, this manifold (or submanifold) $\mathcal{M}$ can be defined implicitly as the vanishing of a smooth function. Given that locally, in a neighbourhood U of a point $a \in \mathcal{M}$, the manifold $\mathcal{M} \subset \mathbb{R}^n$ can be thought of as a surface of dimension d, defined by a given constraint [14],

$$\mathcal{M} = \{y \in U \mid g(y) = 0\}, \tag{1.1}$$

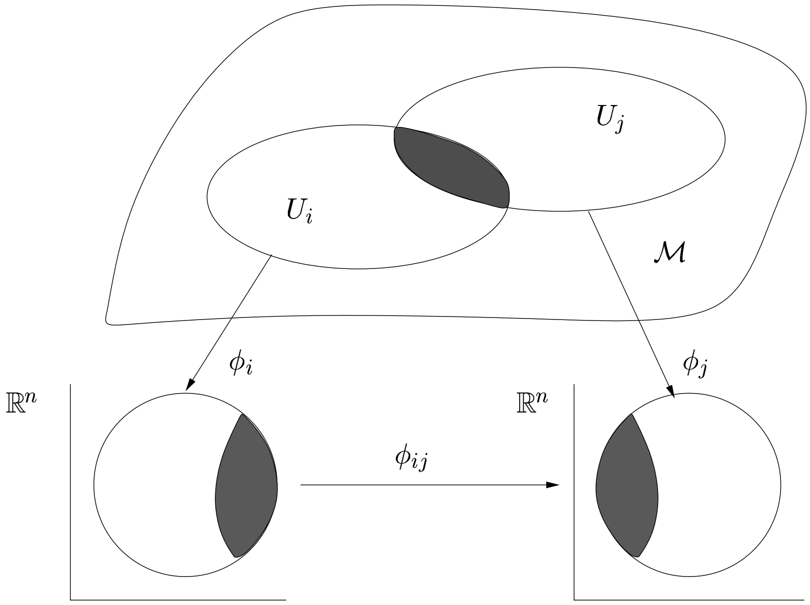


Figure 1.1: Manifold $\mathcal{M}$ with overlapping coordinate charts ϕ_{ij}.

where the constraint $g : U \to \mathbb{R}^m$ is differentiable, $g(a) = 0$, and the rank of $g'(a)$ is equal to $m = n - d$.

Note that in the classical Butcher theory, the manifold $\mathcal{M}$ is defined as the entire n-dimensional real vector space $\mathbb{R}^n$.

An example of using the vanishing function to define a manifold is the constraint for a unit sphere,

$$g(x) = x_1^2 + x_2^2 + x_3^2 - 1,$$

where $x = (x_1, x_2, x_3) \in \mathbb{R}^3$, and $\|x\| = 1$ if and only if $g(x) = 0$.

Another example is the Hamiltonian (energy) $H : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ of a system, which can be thought of as the manifold $\mathcal{M}$ of dimension n. That is, for $p : \mathbb{R} \to \mathbb{R}^n$ and $q : \mathbb{R} \to \mathbb{R}^n$, the vanishing function g is defined as [19],

$$g(p(t), q(t)) = H(p(t), q(t)) - H(p(0), q(0)), \qquad p(t), q(t) \in \mathbb{R}^n, t \in \mathbb{R}.$$

In terms of matrices, another example is to use the orthogonal conditions defining the set of orthogonal matrices $O(n) = \{X \mid X^T X = \mathrm{Id}\}$ of $n(n-1)/2$ dimensions, to define a manifold the vanishing function g of a manifold $\mathcal{M}$ [19],

$$g(X) = \|X^T X - \mathrm{Id}\| = 0,$$

where Id is the $n \times n$ identity matrix.

1.2.2 Tangent Vector, Tangent Bundle and Vector Field

Let $\gamma : \mathbb{R} \to \mathcal{M}$ be a smooth curve on the manifold $\mathcal{M}$, such that $\gamma(0) = p$ on $\mathcal{M}$. Then the vector v at point $p \in \mathcal{M}$ defined as [19],

$$v = \frac{d}{dt}\gamma(t)\bigg|_{t=0}, \qquad \forall t \in \mathbb{R},$$

is called the tangent vector on $\mathcal{M}$ at point $p \in \mathcal{M}$. The set of all tangents at p is called the tangent space of the manifold $\mathcal{M}$ at p, denoted by $T_p\mathcal{M}$. It is the set of equivalence classes of curves at $p \in \mathcal{M}$, given by $[\gamma]_p$ [1]. This means that two curves γ_1 and γ_2 in $[\gamma]_p$ have the same tangent vector v at point p. Furthermore, $T_p\mathcal{M}$ has the structure of a linear vector space. (i.e. if $u, v \in T_p\mathcal{M}$, then $u + v \in T_p\mathcal{M}$ and $\alpha u \in T_p\mathcal{M}$ for $\alpha \in \mathbb{R}$.)

The set of all tangent spaces for all points $p \in \mathcal{M}$ is called the tangent bundle of $\mathcal{M}$, and it is denoted by $T\mathcal{M} = \bigcup_{p\in\mathcal{M}} T_p\mathcal{M}$. This consists of all possible tangent v for any possible base-point $p \in \mathcal{M}$. Also, note that for a d-dimensional manifold $\mathcal{M}$, $T\mathcal{M}$ is a $2d$-dimensional manifold, because we need to specify both v as well as the point p, each of d dimensions respectively.

A vector field F is simply a map $F : \mathcal{M} \to T\mathcal{M}$, such that for a point $p \in \mathcal{M}$, $F(p) \in T_p\mathcal{M}$. The set of all vector fields on $\mathcal{M}$ is denoted $\mathfrak{X}(\mathcal{M})$. Furthermore, addition and multiplication of vector fields are defined in the usual way. That is, for $F, G \in \mathfrak{X}(\mathcal{M})$ and for all $p \in \mathcal{M}$,

$$\begin{aligned} (F+G)(p) &= F(p) + G(p) \\ (\alpha F)(p) &= \alpha(F(p)), \qquad \forall \alpha \in \mathbb{R}. \end{aligned}$$

1.2.3 Differentials, Derivations, and Lie Derivatives

Let $\mathcal{F}(\mathcal{M})$ be the set of smooth, real valued functions on a manifold $\mathcal{M}$. That is,

$$\mathcal{F}(\mathcal{M}) = \{f \mid f : \mathcal{M} \to \mathbb{R}\} .$$

Note that $\mathcal{F}(\mathcal{M})$ has an algebra structure, such that for all $f, g, h \in \mathcal{F}(\mathcal{M})$ and $p \in \mathcal{M}$, we have $(fg)(p) = f(p)g(p)$, $(fg)(p) = (gf)(p)$, and $f(g + h)(p) = (fg)(p) + (fh)(p)$ [1].

A derivation on $\mathcal{F}(\mathcal{M})$ is a mapping $D : \mathcal{F}(\mathcal{M}) \to \mathcal{F}(\mathcal{M})$ with the following properties for all $f, g \in \mathcal{F}(\mathcal{M})$ [3],

R-linear	$D(f+g) = D(f) + D(g)$,	$D(\alpha f) = \alpha D(f)$, $\forall \alpha \in \mathbb{R}$.
Leibniz Rule	$D(fg) = D(f)g + fD(g)$.	
Local operator	$D(f)\vert_{\mathcal{U}} = D(f\vert_{\mathcal{U}})$,	$\forall$ open sets $\mathcal{U} \subset \mathcal{M}$.

We can define a vector field by specifying a derivation operator on $\mathcal{F}(\mathcal{M})$ and vice versa. This is presented as the 'Derivation Theorem' in Chapter 4 of [1].

Theorem 1.1. *If $\mathcal{M}$ is finite dimensional and C^∞, the collection of all derivations on $\mathcal{F}(\mathcal{M})$ is a real vector space isomorphic to $\mathfrak{X}(\mathcal{M})$. In particular, for each derivation D there is a unique $F \in \mathfrak{X}(\mathcal{M})$ such that D is the Lie derivative with respect to $F \in \mathfrak{X}(\mathcal{M})$.*

In Munthe-Kaas's paper [26], Theorem 1.1 is given as a proposition, which states that there is a natural one-to-one correspondence between derivations on $\mathcal{F}(\mathcal{M})$ and vector fields F on $\mathcal{M}$. Furthermore, the derivation corresponding to a given vector field $F \in \mathfrak{X}(\mathcal{M})$ is called the **Lie derivative on $\mathcal{F}(\mathcal{M})$ with respect to F**. The notation is given by $F[\cdot] : \mathcal{F}(\mathcal{M}) \to \mathcal{F}(\mathcal{M})$. In local coordinates $(x^1, \ldots, x^d)$ for a d dimensional manifold $\mathcal{M}$, and vector field $F = \sum_i F^i \frac{\partial}{\partial x^i}$, the Lie derivation on $\mathcal{F}(\mathcal{M})$ with respect to F, when applied to $g \in \mathcal{F}(\mathcal{M})$ is,

$$F[g] = \sum_i F^i \frac{\partial g}{\partial x^i}. \tag{1.2}$$

Similarly, we can consider the bilinear commutator $[\cdot,\cdot] : \mathfrak{X}(\mathcal{M}) \times \mathfrak{X}(\mathcal{M}) \to \mathfrak{X}(\mathcal{M})$, as a derivation on $\mathcal{F}(\mathcal{M})$, such that if $F, G \in \mathfrak{X}(\mathcal{M})$ and for all $f \in \mathcal{F}(\mathcal{M})$, we have,

$$[F, G](f) = F[G[f]] - G[F[f]].$$

However, if $F \in \mathfrak{X}(\mathcal{M})$ is fixed, then the map $[F,\cdot] : \mathfrak{X}(\mathcal{M}) \to \mathfrak{X}(\mathcal{M})$ is a derivation on $\mathfrak{X}(\mathcal{M})$. This is the **Lie derivative on $\mathfrak{X}(\mathcal{M})$ with respect to F** [26]. In local coordinates, if the two vector fields F and G are given by,

$$F = \sum_i F^i \frac{\partial}{\partial x^i}, \qquad \text{and} \qquad G = \sum_j G^j \frac{\partial}{\partial x^j},$$

then the Lie derivative with respect to F is given by,

$$[F, G] = FG - GF = \sum_{i,j} \left(F^j \frac{\partial G^i}{\partial x^j} - G^j \frac{\partial F^i}{\partial x^j} \right) \frac{\partial}{\partial x^i}, \qquad \forall G \in \mathfrak{X}(\mathcal{M}). \tag{1.3}$$

In this case, F is both a vector field and a derivation operator. The vector field $[F, G]$ is often referred to as the Lie (Jacobi) Bracket [1].

The **differential** of $g \in \mathcal{F}(\mathcal{M})$ is denoted $\mathbf{d}g$. This is the derivative of the local representation of g, such that in local coordinates, the components of $\mathbf{d}g$ is given by,

$$(\mathbf{d}g)_i = \frac{\partial g}{\partial x^i}.$$

This means that we can rewrite the expression for the Lie derivative on $\mathcal{F}(\mathcal{M})$ with respect to $F \in \mathfrak{X}(\mathcal{M})$ in Eq (1.2) as,

$$F[g] = \sum_i F^i \frac{\partial g}{\partial x^i} = \sum_i F^i (\mathbf{d}g)_i = \mathbf{d}g(F), \qquad \forall g \in \mathcal{F}(\mathcal{M}). \tag{1.4}$$

1.2.4 Integral Curves and Flow

Let $F \in \mathfrak{X}(\mathcal{M})$, and $y : \mathbb{R} \to \mathcal{M}$ be a curve on $\mathcal{M}$ such that at $t = 0$, $y(0) = p \in \mathcal{M}$.

Definition 1.3. *An integral curve of the vector field $F : \mathcal{M} \to T\mathcal{M}$ at a point $p \in \mathcal{M}$ is a curve $y : \mathbb{R} \to \mathcal{M}$ with $y(0) = p$, such that [1],*

$$\frac{d}{dt} y = F(y), \qquad y(0) = p \in \mathcal{M}.$$

Note that dy/dt is a curve from $\mathbb{R} \to T\mathcal{M}$.

Definition 1.4. *A differential equation evolving on $\mathcal{M}$ is a differential equation of the form [19],*

$$y' = F(y), \qquad t \geq 0, \text{ and } y(0) = p \in \mathcal{M},$$

where $F \in \mathfrak{X}(\mathcal{M})$. The flow of F is the map $\Psi_{F,t} : \mathcal{M} \to \mathcal{M}$, such that,

$$y(t) = \Psi_{F,t}(y(0)) = \Psi_{F,t}(p), \qquad \forall t \in \mathbb{R},$$

is the solution to the differential equation evolving on $\mathcal{M}$.

Note that the flow of F is defined in such a way that as t varies, $y(t) = \Psi_{F,t}(p)$ is the integral curve which starts at $y(0) = p \in \mathcal{M}$ [27],

$$F(\Psi_{F,0}(p)) = \frac{d}{dt} \Psi_{F,t}(p) \bigg|_{t=0}, \qquad \Psi_{F,0}(p) = p.$$

Note that $F(\Psi_{F,0}(p)) \in T_p\mathcal{M}$, and the vector field F is often referred to as the **infinitesimal generator of the flow $\Psi_{F,t}$**.

This flow operator $\Psi_{F,t}$ satisfies the following identity [19],

$$\Psi_{F,\alpha} = \Psi_{\alpha F,1},$$

which can be proved with the scaling of time t by a constant α, such that $y(\alpha t) = \Psi_{F,\alpha t}(y_0)$ is the solution to the initial value problem, for $y_0 \in \mathcal{M}$,

$$y'(\alpha t) = F(y(\alpha t)), \quad y(0) = y_0.$$

But if we have a change of variable such that $z(t) = y(\alpha t)$, then $z(t)$ is the solution to the initial value problem,

$$z'(t) = y'(\alpha t) = \alpha y'(\alpha t) = \alpha F(y(\alpha t)) = \alpha F(z(t)),$$

for $z(0) = y(0) = y_0$. So $z(t)$ has the flow $z(t) = \Psi_{\alpha F,t}(y_0)$, and given that $y(\alpha t) = z(t)$,

$$\Psi_{F,\alpha t}(y_0) = \Psi_{\alpha F,t}(y_0).$$

So for $t = 1$, we have the identity given above, since the equation holds for all $y_0 \in \mathcal{M}$.

1.2.5 Push-forward and Pull-back Maps

Let $\phi : \mathcal{M} \to \mathcal{N}$ be a diffeomorphism between two manifolds $\mathcal{M}$ and $\mathcal{N}$. We can talk about the push-forward and pull-back maps between the two manifolds $\mathcal{M}$ and $\mathcal{N}$, according to the following definitions below. Let $f \in \mathcal{F}(\mathcal{M})$, $g \in \mathcal{F}(\mathcal{N})$, ϕ^{-1} is the inverse of the mapping ϕ, and $\phi' : T\mathcal{M} \to T\mathcal{N}$ is the derivative of ϕ.

Definition 1.5. *The push-forward and pull-back of a function or a vector field is defined in the following way [1]:*

- *The push-forward of a function $f \in \mathcal{F}(\mathcal{M})$ along ϕ is given by the map $(\phi_* f) : \mathcal{N} \to \mathbb{R}$, such that for a point $y \in \mathcal{N}$,*

$$(\phi_* f)(y) = (f \circ \phi^{-1})(y). \tag{1.5}$$

- *The pull-back of a function $g \in \mathcal{F}(\mathcal{N})$ along ϕ is given by $(\phi^* g) : \mathcal{M} \to \mathbb{R}$, such that for a point $x \in \mathcal{M}$,*

$$\phi^* g(x) = (g \circ \phi)(x). \tag{1.6}$$

- *The push-forward of a vector field $X \in \mathfrak{X}(\mathcal{M})$ along ϕ is given by $(\phi_* X) : \mathcal{N} \to T\mathcal{N}$, such that for a point $y \in \mathcal{N}$,*

$$(\phi_* X)(y) = (\phi' \circ X \circ \phi^{-1})(y). \tag{1.7}$$

- *The pull-back of a vector field $Y \in \mathfrak{X}(\mathcal{N})$ along ϕ is given by $(\phi^* Y) : \mathcal{M} \to T\mathcal{M}$, such that for a point $x \in \mathcal{M}$,*

$$\begin{aligned}(\phi^* Y)(x) &= ((\phi^{-1})_* Y)(x) \\ &= ((\phi^{-1})' \circ Y \circ \phi)(x),\end{aligned} \tag{1.8}$$

  where $\phi^* = (\phi^{-1})_*$.

$$\begin{tikzcd}
T\mathcal{M} \arrow[r, "\phi' : T\mathcal{M} \to T\mathcal{N}"] & T\mathcal{N} \\
\mathcal{M} \arrow[u, "X \in \mathfrak{X}(\mathcal{M})"] \arrow[r, "\phi : \mathcal{M} \to \mathcal{N}"'] \arrow[ur, phantom, "\phi' \circ X = Y \circ \phi"] & \mathcal{N} \arrow[u, "Y \in \mathfrak{X}(\mathcal{N})"']
\end{tikzcd}$$

Figure 1.2: ϕ-relatedness.

Note that the concept of pull-backs allows for the translation of two tangent vectors at different points on $\mathcal{M}$ to a single common point. If $\mathcal{M}$ has local coordinates x^i and $\mathcal{N}$ has local coordinates y^j, then the components of the push-forward $\phi_* X$ is [1],

$$(\phi_* X)^i(y) = \frac{\partial \phi^i}{\partial x^j} X^j(\phi^{-1}(y)) = \frac{\partial \phi^i}{\partial x^j} X^j(x).$$

Furthermore, if the map ϕ is not a diffeomorphism, we can still talk about the 'relatedness' of the two vector fields, such that $X \in \mathfrak{X}(\mathcal{M})$ and $Y \in \mathfrak{X}(\mathcal{N})$ are ϕ-related if,

$$\phi' \circ X = Y \circ \phi. \tag{1.9}$$

This is denoted by $X \sim_\phi Y$, and leads to the commutivity in Figure 1.2 from [1]. Note that if ϕ is a diffeomorphism, then $X \sim_\phi Y$ is equivalent to saying that X is the pull-back of Y. The relatedness of two vector fields will be discussed further in later chapters.

Figure (1.3) shows how the push-forward map of ϕ works on a vector field $X \in \mathfrak{X}(\mathcal{M})$, and on the flow Ψ_t of X. Note that $\phi \circ \Psi_t \circ \phi^{-1}$ is the push-forward of flow Ψ_t by map ϕ. It also implies that the flow of the push-forward of a vector field is the push-forward of its flow [1].

1.2.6 Relationship between Lie Derivatives and Pull-backs.

In [26], the pull-back of a vector field $Y \in \mathfrak{X}(\mathcal{N})$, acting as the derivation operator on the pull-back of a function $g \in \mathcal{F}(\mathcal{N})$ is defined as,

$$(\phi^* Y)[\phi^* g] = \phi^*(Y[g]), \qquad \text{where } \phi : \mathcal{M} \to \mathcal{N}. \tag{1.10}$$

Note that $\phi^* g : \mathcal{M} \to \mathbb{R}$, so $\phi^* g \in \mathcal{F}(\mathcal{M})$, while the pull-back of the vector field $Y \in \mathfrak{X}(\mathcal{N})$, is denoted by the map $\phi^* Y : \mathcal{M} \to T\mathcal{M}$, and in this case, it is also the derivation operator $(\phi^* Y)[\cdot] : \mathcal{F}(\mathcal{M}) \to \mathcal{F}(\mathcal{M})$.

Furthermore, there are two relations between Lie derivatives on $\mathcal{F}(\mathcal{M})$, $\mathfrak{X}(\mathcal{M})$, and pull-backs. They arise because we can replace $\phi : \mathcal{M} \to \mathcal{N}$ with the flow $\Psi_{F,t} : \mathcal{M} \to \mathcal{M}$ of a given fixed vector field $F \in \mathfrak{X}(\mathcal{M})$. Recall that the Lie derivative on $\mathcal{F}(\mathcal{M})$ with respect to $F \in \mathfrak{X}(\mathcal{M})$ is the map $F[\cdot] : \mathcal{F}(\mathcal{M}) \to \mathcal{F}(\mathcal{M})$, and the Lie derivative on $\mathfrak{X}(\mathcal{M})$ with respect to $F \in \mathfrak{X}(\mathcal{M})$ is $[F, \cdot] : \mathfrak{X}(\mathcal{M}) \to \mathfrak{X}(\mathcal{M})$. Then we have the following lemma.

Lemma 1.1. *For $f \in \mathcal{F}(\mathcal{M})$ and $G \in \mathfrak{X}(\mathcal{M})$, the Lie derivatives $F[\cdot]$ and $[F, \cdot]$ are related to the pull-backs of the flow $\Psi_{F,t}$ by the following rates of change, for all $t \in \mathbb{R}$ [1],*

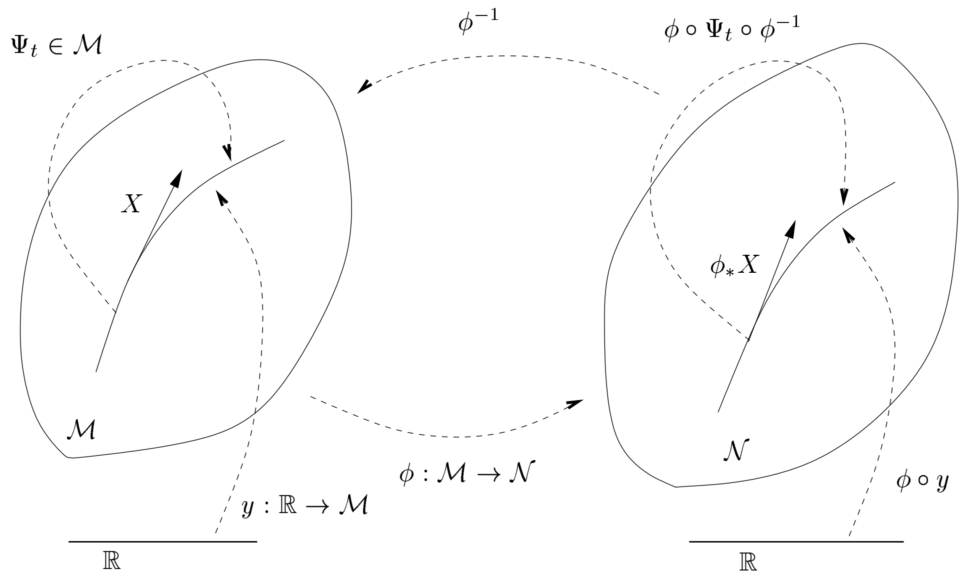


Figure 1.3: Push-forward map.

$$\frac{d}{dt}\left(\Psi^*_{F,t} f\right) = \Psi^*_{F,t} F[f], \tag{1.11}$$

$$\frac{d}{dt}\left(\Psi^*_{F,t} G\right) = \Psi^*_{F,t}[F, G]. \tag{1.12}$$

Proof. To prove Eq (1.11), for $p \in \mathcal{M}$, $f \in \mathcal{F}(\mathcal{M})$, and $F \in \mathfrak{X}(\mathcal{M})$ we have,

$$\begin{aligned} \frac{d}{dt}(\Psi^*_{F,t} f)(p) &= \frac{d}{dt}(f \circ \Psi_{F,t})(p) = \mathbf{d}f(\Psi_{F,t}(p))\frac{d}{dt}\Psi_{F,t}(p) \\ &= \mathbf{d}f(\Psi_{F,t}(p)) \cdot F(\Psi_{F,t}(p)) = \mathbf{d}f(F)(\Psi_{F,t}(p)) \\ &= F[f](\Psi_{F,t}(p)) = \Psi^*_{F,t}(F[f])(p). \end{aligned}$$

To prove Eq (1.12), since G is a vector field, then by Theorem 1.1, the one-to-one correspondence between vector field and derivation means that it is sufficient to prove that both sides of Eq (1.12) are equal when acting on an arbitrary function $f \in \mathcal{F}(\mathcal{M})$ [1]. So, from the left hand side of Eq (1.12), at $t = 0$, we have,

$$\begin{aligned} \tfrac{d}{dt}\Psi^*_{F,t} G[f](p)\Big|_{t=0} &= \tfrac{d}{dt}\Big|_{t=0}\Psi^*_{F,t} G[\Psi^*_{F,t}(\Psi^*_{F,-t} f)](p) && \text{by Eq (1.10)} \\ &= \tfrac{d}{dt}\Big|_{t=0} \Psi^*_{F,t}(G[\Psi^*_{F,-t} f])(p) \\ &= \Psi^*_{F,t} F[G[\Psi^*_{F,-t} f]](p)\Big|_{t=0} + G[\Psi^*_{F,t} F[\Psi^*_{F,-t} f]](p)\Big|_{t=0} && \text{by Eq (1.11)} \\ &= \Psi^*_{F,t} F[G[\Psi^*_{F,-t} f]](p)\Big|_{t=0} + G[\Psi^*_{F,t} F[-\Psi^*_{F,t} f]](p)\Big|_{t=0} && \text{and Leibniz Rule} \\ &= \Psi^*_{F,t} F[G[\Psi^*_{F,-t} f]](p)\Big|_{t=0} + G[\Psi^*_{F,t}(F[-f])](p)\Big|_{t=0} && \text{by Eq (1.10)} \\ &= F[G[f]](p) + G[F[-f]](p) \\ &= F[G[f]](p) - G[F[f]](p) \\ &= [F, G][f](p). \end{aligned}$$

From the right hand side of Eq (1.12), at $t = 0$, we have,

$$\Psi^*_{F,t}[F,G][f](p)\Big|_{t=0} = [F,G][f](p).$$

Since both sides of Eq (1.12) acting on $f \in \mathcal{F}(\mathcal{M})$ as a derivation operator yield the same result, then Eq (1.12) is correct. □

Note that for time dependent $F(t), G(t) \in \mathfrak{X}(\mathcal{M})$, and $f(t) \in \mathcal{F}(\mathcal{M})$, Eq (1.11) and Eq (1.12) become,

$$\begin{aligned} \frac{d}{dt}\Big(\Psi^*_{F(t),t} f(t)\Big) &= \Psi^*_{F(t),t}\Big(F(t)[f(t)] + \frac{\partial}{\partial t} f(t)\Big), & (1.13) \\ \frac{d}{dt}\Big(\Psi^*_{F(t),t} G(t)\Big) &= \Psi^*_{F(t),t}\Big([F(t),G(t)] + \frac{\partial}{\partial t} G(t)\Big). & (1.14) \end{aligned}$$

For their proofs, please refer to Chapter 4 of [1]. Also, note that when $t = 0$, Eq (1.11) reduces to,

$$\frac{d}{dt}(\Psi^*_{F,t} f)\Big|_{t=0} = F[f]. \qquad (1.15)$$

If we use Taylor's expansion on $\Psi^*_{F,t} f$ around $t = 0$, and apply Eq (1.15), we find,

$$\begin{aligned} \Psi^*_{F,t} f &= \Psi^*_{F,t} f\Big|_{t=0} + t\left[\frac{\partial}{\partial t}\Psi^*_{F,t} f\right]\Big|_{t=0} + \frac{t^2}{2!}\left[\frac{\partial^2}{\partial t^2}\Psi^*_{F,t} f\right]\Big|_{t=0} + \frac{t^3}{3!}\left[\frac{\partial^3}{\partial t^3}\Psi^*_{F,t} f\right]\Big|_{t=0} + \cdots \\ &= f + tF[f] + \frac{t^2}{2}F[F[f]] + \frac{t^3}{3!}F[F[F[f]]] + \cdots \\ &= \sum_{k=0}^{\infty} \frac{t^k}{k!} F^k[f]. & (1.16) \end{aligned}$$

This is known as the **Lie series on manifolds**, and it will be used to derive the Lie-Butcher series in Chapter 3. This also leads to the proof for the order conditions associated with the Runge-Kutta-Munthe-Kaas methods in Chapter 3.

1.3 Lie Group and Lie Algebra

A Lie group G and its associated Lie Algebra $\mathfrak{g}$ are the building blocks in which the whole theory for Lie group methods in geometric integration is based upon. In the following sections, we will set up some of the terminologies used in the later chapters.

1.3.1 Lie Group G

A group is a set with a group operation, like multiplication, such that if $a, b \in G$, then their product is also in G. This group operation must be associative. The group G must contain an identity element e, and every element in G must also have an inverse. Similarly, a Lie group G is a differentiable manifold (i.e. a group which carries the structure of a differentiable manifold), so their elements can be continuously varied. Since a Lie group G is a group, then there is a group operation, $G \times G \to G$, satisfying the following properties [19],

Associativity	$g \cdot (h \cdot k) = (g \cdot h) \cdot k,$	$\forall g, h, k \in G.$
Identity element $e \in G$	$e \cdot g = g = g \cdot e,$	$\forall g \in G.$
Inverse $g^{-1} \in G$	$g^{-1} \cdot g = g \cdot g^{-1} = e,$	$\forall g \in G.$

The product for a Lie group G is defined in two ways for all elements $a, b \in G\ \cdot$, by the 'left' and 'right' multiplications in the group,

$$L_a(b) = a \cdot b, \qquad R_a(b) = b \cdot a \qquad \forall a, b \in G.$$

Definition 1.6. *Let $\mathcal{M}$ be a differentiable manifold and G be the Lie group. A 'left' Lie group action on $\mathcal{M}$ is a smooth mapping $\Lambda : G \times \mathcal{M} \to \mathcal{M}$ such that [28],*

$$\begin{aligned} \Lambda(e, p) &= e \cdot p = p, && \forall p \in \mathcal{M} \text{ and the identity element } e \in G, \\ \Lambda(g_1 \cdot g_2, p) &= \Lambda(g_1, \Lambda(g_2, p)), && \forall p \in \mathcal{M}, \text{ and } g_1, g_2 \in G. \end{aligned} \tag{1.17}$$

Note that $\Lambda : G \times \mathcal{M} \to \mathcal{M}$ is a local Lie group action if the above relations only hold for elements in the neighbourhood of the identity $e \in G$. Furthermore, Λ is a transitive action on $\mathcal{M}$, if for every pair $m_1, m_2 \in \mathcal{M}$, there exists at least one element $g \in G$ such that $\Lambda(g, m_1) = m_2$ [29].

Definition 1.7. *A homogeneous space is a manifold with a transitive Lie group action [29].*

In most cases we deal with linear Lie groups, that is, groups with elements which are linear operators on some vector space V. We can represent these linear operators by matrices. Consider the **general linear Lie group** $GL(V)$, the group of linear operators on the real n-dimensional vector space V which are bijective. We define a matrix representation of these operators by taking a basis in the vector space V, and by considering the action of the operators on the basis vectors [3].

Let $\{e_1, e_2, \dots, e_n\}$ be a basis in vector space V. Then the $n \times n$ matrix (a_{ij}) representing the linear operator $A \in GL(V)$ is defined by,

$$Ae_i = \sum_{j=1}^{n} e_j a_{ji}, \qquad \text{for } i = 1, \dots, n.$$

In this way, the general linear Lie group $GL(V)$ is isomorphic to the **general linear matrix Lie group $\boldsymbol{GL(n)}$**. $GL(n)$ is a group of all invertible $n \times n$ matrices, that is, $GL(n) = \{A \mid \det(A) \neq 0\}$. The group operation is matrix multiplication, and the identity element is given by the $n \times n$ identity matrix Id.

Examples of matrix Lie groups include the special linear matrix Lie group $SL(n)$, the orthogonal matrix Lie group $O(n)$, and the symplectic matrix Lie group $Sp(2k)$. They are called the **classical matrix Lie groups**, and are the subgroups of the general linear matrix Lie group $GL(n)$. The following section is a brief discussion of some of the matrix Lie groups and their properties.

The subset of linear operators $A \in GL(n)$ with $\det(A) = 1$ defines the **special linear matrix Lie group $\boldsymbol{SL(n)}$**, that is, $SL(n) = \{A \in GL(n) | \det(A) = 1\}$.

The **orthogonal matrix Lie group $\boldsymbol{O(n)}$** is given by $O(n) = \{A \in GL(n) \mid A^T A = \mathrm{Id}\}$. With respect to the orthonormal basis $\{e_1, e_2, \dots, e_n\}$, of the n-dimensional real vector space V, there is a bilinear form, given by,

$$\langle e_i, e_j \rangle = \delta_{ij}, \qquad \text{for } i, j = 1, \dots, n.$$

The vectors $x, y \in V$ then have the following decompositions, for $x_i, y_i \in \mathbb{R}$,

$$x = \sum_{i=1}^{n} x_i e_i, \quad \text{and} \quad y = \sum_{i=1}^{n} y_i e_i,$$

and

$$\langle x, y \rangle = \sum_{ij} x_i y_j \langle e_i, e_j \rangle = \sum_{ij} x_i y_i \delta_{ij} = \sum_{i} x_i y_i = x^T y = y^T x. \tag{1.18}$$

The subset of operators $A \in GL(n)$ which leave Eq (1.18) invariant is the orthogonal Lie group. We can consider the matrix representation of the linear operators by their actions, with respect to the orthonormal basis $\{e_1, \ldots, e_n\}$. Given that $A \in GL(n)$ leaves Eq (1.18) invariant, then for all $x, y \in V$ we have,

$$\begin{aligned} \langle x, y \rangle &= \langle Ax, Ay \rangle \\ \Rightarrow \langle x, y \rangle &= x^T y = (Ax)^T (Ay) = x^T A^T A y. \end{aligned}$$

The above is true if and only if A is an orthogonal matrix,

$$A^T A = \mathrm{Id}. \tag{1.19}$$

So, the orthogonal matrix Lie group is given by $O(n) = \{A \in GL(n) | A^T A = \mathrm{Id}\}$.

Furthermore from Eq (1.19), we have,

$$1 = \det(\mathrm{Id}) = \det(A^T A) = \det(A^T) \det(A) = \det(A)^2.$$

Hence, the determinant of orthogonal matrices equals ± 1. The subset of matrices $A \in O(n)$ with $\det(A) = 1$ is a subgroup of $O(n)$ called the **special orthogonal matrix Lie group**, $SO(n) = \{A \in O(n) | \det(A) = 1\}$.

To define the **symplectic matrix Lie group $Sp(2k)$**, we start with a $2k$-dimensional real vector space V, with a bilinear skew-symmetric form,

$$\langle x, y \rangle = -\langle y, x \rangle, \qquad \forall x, y \in V.$$

One can prove that V has a basis $\{e_1, \ldots, e_{2k}\}$ such that [3],

$$\langle e_i, e_{j+k} \rangle = \delta_{ij}, \qquad \forall i, j = 1, \ldots, k,$$

and the skew-symmetry implies,

$$\langle e_{j+k}, e_i \rangle = -\langle e_i, e_{j+k} \rangle = -\delta_{ij}, \qquad \forall i, j = 1, \ldots, k.$$

If $x, y \in V$, then with respect to the basis defined above, x, y have the decompositions, for $x_i, y_i \in \mathbb{R}$ [3],

$$x = \sum_{i=1}^{k} x_i e_i + \sum_{i=1}^{k} x_{i+k} e_{i+k}, \qquad y = \sum_{i=1}^{k} y_i e_i + \sum_{i=1}^{k} y_{i+k} e_{i+k},$$

and the bilinear form is given by,

$$\begin{aligned} \langle x, y \rangle &= \sum_{ij} x_i y_{j+k} \langle e_i, e_{j+k} \rangle + \sum_{ij} x_{i+k} y_j \langle e_{i+k}, e_j \rangle \\ &= \sum_{ij} x_i y_{j+k} \delta_{ij} - \sum_{ij} x_{i+k} y_j \delta_{ij} \\ &= \sum_{i} x_i y_{i+k} - \sum_{i} x_{i+k} y_i = x^T J y, \end{aligned} \tag{1.20}$$

where J is a matrix of the form,

$$J = \begin{bmatrix} 0 & \mathrm{Id}_{k\times k} \\ -\mathrm{Id}_{k\times k} & 0 \end{bmatrix}. \tag{1.21}$$

The subset of linear operators $A \in GL(V)$ which leave Eq (1.20) invariant is the symplectic Lie group $Sp(V)$. Using the basis $\{e_1, \ldots, e_{2k}\}$ defined for the real symplectic vector space V, the matrix representation of this symplectic matrix Lie group $Sp(2k)$ must satisfy,

$$\begin{aligned} \langle x, y\rangle &= \langle Ax, Ay\rangle \qquad \forall x, y \in V \\ \Rightarrow \langle x, y\rangle &= x^T J y = (Ax)^T J(Ay) = x^T A^T J A y. \end{aligned}$$

The above is true if and only if matrix A satisfies,

$$A^T J A = J, \qquad \text{where } J \text{ is given by Eq (1.21).} \tag{1.22}$$

Hence, the symplectic matrix Lie group is defined by,

$$Sp(2k) = \left\{ A \in GL(2k) \mid A^T J A = J \right\}.$$

1.3.2 Lie Algebra $\mathfrak{g}$

Given a particular Lie group G, it has an associative Lie algebra $\mathfrak{g} = T_e G$, which is the tangent space at the identity e of the Lie group G. In particular, if G is finite-dimensional, then its Lie algebra $\mathfrak{g}$ is a linear space of the same dimension. Also, $\mathfrak{g}$ is closed under the bilinear, skew-symmetric operation given by the Lie Bracket, $[\cdot,\cdot] : \mathfrak{g} \times \mathfrak{g} \to \mathfrak{g}$, defined as,

$$[u, v] = uv - vu, \qquad \forall u, v \in \mathfrak{g}. \tag{1.23}$$

Alternatively, in terms of two smooth curves $\rho : \mathbb{R} \to G$, $\sigma : \mathbb{R} \to G$ such that at $s = 0$ and $t = 0$, $\rho(0) = \sigma(0) = e$ and their derivatives $\rho'(0) = u$ and $\sigma'(0) = v$, the Lie bracket $[\cdot,\cdot] : \mathfrak{g} \times \mathfrak{g} \to \mathfrak{g}$ is given by [19],

$$[u, v] = \frac{\partial^2}{\partial s \partial t} \rho(s)\sigma(t)\rho(-s)\bigg|_{s=t=0}. \tag{1.24}$$

The proofs for the above can be found in Section 1.3.3.

This Lie bracket must also satisfy the following axioms for all u, v, $w \in \mathfrak{g}$, and $\alpha, \beta \in \mathbb{R}$ as constants [30](pg43),

$$\begin{aligned} [\alpha u + \beta v, w] &= \alpha [u, w] + \beta [v, w], \\ [u, v] &= -[v, u], \\ [u, [v, w]] + [v, [w, u]] + [w, [u, v]] &= 0. \end{aligned}$$

Elements of $\mathfrak{g}$ are uniquely determined by their values at the identity $e \in G$. Recall that we have the 'right' multiplication, $R_a(b) = b \cdot a \in G$, for all $a, b \in G$. To denote the same operation in the tangent space, say $T_a G$ for every $a \in G$, we have the mapping [27],

$$R'_a = T_e R_a : \mathfrak{g} \to T_a G.$$

For $a \in G$, $v \in \mathfrak{g} = T_e G$, this operation is given by,

$$R'_a(v) = v \cdot a. \tag{1.25}$$

Hence, the tangent space at any point $a \in G$ can be identified with $\mathfrak{g} = T_eG$ using the notation R'_a, for all $a \in G$ [27],

$$T_aG = \left\{R'_a(v) \mid v \in \mathfrak{g}, \right\},$$

and for all $v \in \mathfrak{g}$ and $a \in \mathcal{M}$, we can denote a vector field of the form $X_v(a) = R'_a(v)$ as a **right invariant vector field**.

Another important idea is the action of a Lie algebra on $\mathcal{M}$, denoted by a smooth function $\lambda : \mathfrak{g} \times \mathcal{M} \to \mathcal{M}$, where its definition can be found in Definition 1.12. Each fixed element $v \in \mathfrak{g}$ generates a vector field on $\mathcal{M}$, denoted $(\lambda_* v) : \mathcal{M} \to T\mathcal{M}$ and defined by [28],

$$(\lambda_* v)(p) = \frac{d}{dt}\lambda(tv, p)\bigg|_{t=0}, \qquad \forall p \in \mathcal{M}. \tag{1.26}$$

Note that since $v \in \mathfrak{g}$ corresponds to the vector field $\lambda_*(v) \in \mathfrak{X}(\mathcal{M})$, and vector field is also known as the infinitesimal generator from Section 1.2.4, then the **infinitesimal generator of the action** corresponding to $v \in \mathfrak{g}$ is denoted by [11],

$$v_{\mathcal{M}} = (\lambda_* v).$$

The above notation was used by Munthe-Kaas in his discussion of the Runge-Kutta-Munthe-Kaas (RKMK) theory in [28]. For more details, please refer to Section 1.3.3 and Chapter 3 of this thesis.

Now, $(\lambda_* v)$ is a vector field in $\mathfrak{X}(\mathcal{M})$, and thus $(\lambda_* v)(p) \in T_p\mathcal{M}$ for all $p \in \mathcal{M}$. Therefore, for all $p \in \mathcal{M}$, $(\lambda_*(\cdot))(p) : \mathfrak{g} \to T_p\mathcal{M}$. If $\mathcal{M} = G$, then $(\lambda_*(\cdot))(p)$ in Eq (1.26) is the same as the notation $R'_p(\cdot) : \mathfrak{g} \to T_p\mathcal{M}$ introduced at the beginning of this section. Hence, differential equations evolving on $\mathcal{M}$, with respect to some point $p \in \mathcal{M}$ in the tangent space $T_p\mathcal{M}$, can be rewritten as infinitesimal generators of the action λ. Such differential equations given by the infinitesimal generator $v_{\mathcal{M}}$ for some $v \in \mathfrak{g}$, are presented in their generic forms [28]. This concept is important if we are to use the RKMK method to solve the differential equations on $\mathcal{M}$. More information can be found in Section 1.3.5 or from [28].

Now we want to consider the concept of a Lie algebra homomorphism, which leads to the famous theorem called Ado's Theorem connecting the general Lie theory with the matrix representations of Lie algebras [3].

Definition 1.8. *For two Lie algebras $\mathfrak{g}$ and $\mathfrak{h}$, a mapping $\phi : \mathfrak{g} \to \mathfrak{h}$ is called a* **Lie algebra homomorphism** *if,*

1. *$\phi : \mathfrak{g} \to \mathfrak{h}$ is a linear map, such that for $\alpha, \beta \in \mathbb{R}$ and for all $u, v \in \mathfrak{g}$, we have,*

$$\phi(\alpha u + \beta v) = \alpha\phi(u) + \beta\phi(v).$$

2. *For all $u, v \in \mathfrak{g}$ and the Lie bracket $[\cdot,\cdot]$, we have,*

$$\phi([u, v]) = [\phi(u), \phi(v)].$$

Definition 1.9. *Two Lie algebras $\mathfrak{g}$ and $\mathfrak{h}$ are* **isomorphic** *($\mathfrak{g} \cong \mathfrak{h}$) when a homomorphism $\phi : \mathfrak{g} \to \mathfrak{h}$ is bijective (i.e. one-to-one and onto). In other words, two Lie algebras are isomorphic when the homomorphism is invertible.*

Definition 1.10. *An isomorphism in* $\phi : \mathfrak{g} \to \mathfrak{g}$ *is called an* **automorphism**.

In [3], the concept of a representation for a Lie algebra is defined in terms of a Lie algebra homomorphism. Note that the definitions for $\mathfrak{gl}(n)$ and $\mathfrak{gl}(V)$ are given later.

Definition 1.11. *A* **(linear) representation** ρ *of a Lie algebra* $\mathfrak{g}$ *is a homomorphism* ρ *of* $\mathfrak{g}$ *into the general linear Lie algebra* $\mathfrak{gl}(V)$ *of linear operators on a vector space* V. *A* **matrix representation** *is a homomorphism* M *of* $\mathfrak{g}$ *into a matrix Lie algebra* $\mathfrak{gl}(n)$ *of* $n \times n$ *matrices.*

Let ρ be a representation of $\mathfrak{g}$. Then every element $u \in \mathfrak{g}$ is mapped to (represented by) a linear operator $\rho(u)$ acting on V, and the homomorphism ρ satisfies Definition 1.8. So, for all $u, v \in \mathfrak{g}$ and all $\alpha, \beta \in \mathbb{R}$,

$$\begin{aligned} \rho(\alpha u + \beta v) &= \alpha\rho(u) + \beta\rho(v), \quad \text{where } \rho(u), \rho(v) \in \mathfrak{gl}(V), && (1.27) \\ \rho([u, v]) &= [\rho(u), \rho(v)]. && (1.28) \end{aligned}$$

When the operators $\rho(u)$ for $u \in \mathfrak{g}$ are matrices, then the representation is a matrix representation. This matrix representation is a map $M : \mathfrak{g} \to \mathfrak{gl}(n)$ such that,

$$M(u) \in \mathfrak{gl}(n), \qquad \forall u \in \mathfrak{g},$$

for which the relationships in Eq (1.27) and Eq (1.28) still hold when ρ is replaced by the matrix operator M.

Note that when the vector space V is finite dimensional, then choosing a basis in V turns a representation ρ into a matrix representation M and vice versa. This is presented more formally in **Ado's Theorem** [19].

Theorem 1.2 (Ado's Theorem). *Every finite-dimensional Lie algebra is isomorphic to a subalgebra of the matrix algebra* $\mathfrak{gl}(n)$ *for some* $n \geq 1$.

Note that a similar result does not hold for all finite-dimensional Lie groups, but it is true for the finite-dimensional local Lie groups. That is, given a finite-dimensional local Lie group action, we can find an equivalent local action by a matrix Lie group. So, given that Theorem 1.2 is true for all finite-dimensional Lie algebras, and finite-dimensional local Lie groups, then the above can be summarised in the following way, as quoted from [19]:

> *For practically any concept in general Lie theory, there exists a corresponding concept within matrix Lie theory. Vice versa, practically any result that holds in the matrix case remains valid within the general Lie theory.*

Ado's Theorem provides a connection between the abstract Lie theory and their matrix representations. This can be demonstrated more clearly by looking at some of the examples of Lie algebras below.

To introduce the **general linear Lie algebra** $\mathfrak{gl}(V)$, we first consider the set of all linear operators on the finite-dimensional real vector space V [3]. The multiplication and addition of these linear operators are defined in the usual way, for all $\alpha, \beta \in \mathbb{R}$, $u, v \in \mathfrak{gl}(V)$, and the identity $e \in V$,

$$\begin{aligned} (\alpha u + \beta v)e &= \alpha(ue) + \beta(ve), \\ (uv)e &= u(ve), \end{aligned}$$

along with the bilinear bracket operation given by the commutator in Eq (1.23),

$$[u, v] = uv - vu.$$

As with the discussion on Lie groups and their matrix representations in Section 1.3.1, we can talk about the **general linear matrix Lie algebra $\mathfrak{gl}(n)$** with respect to their actions on a basis $\{e_1, \ldots, e_n\}$ in the n-dimensional vector space V. In this vector space, every linear operator u can be represented by a $n \times n$ matrix $u = u_{ij}$ such that,

$$ue_i = \sum_{k=1}^{n} e_k u_{ki}, \qquad \text{for } i = 1, \ldots, n.$$

The set of $n \times n$ matrices obtained in this way is a n^2-dimensional vector space. Defining a commutator similarly to Eq (1.23), and taking the matrix multiplication as the product operation on V, we have the general linear matrix Lie algebra $\mathfrak{gl}(n)$. Note that by Ado's Theorem, $\mathfrak{gl}(n)$ is isomorphic to $\mathfrak{gl}(V)$.

Examples of the matrix Lie algebras corresponding to the classical matrix Lie groups include the special linear matrix Lie algebra $\mathfrak{sl}(n)$, the orthogonal matrix Lie algebra $\mathfrak{o}(n)$, and the symplectic matrix Lie algebra $\mathfrak{sp}(2k)$. They are discussed in more detail in the following section.

The proof for the condition on the **special linear matrix Lie algebra $\mathfrak{sl}(n)$** is based on the identity for the matrix operator $M \in \mathfrak{gl}(n)$,

$$\det(\exp(tM)) = \exp(\mathrm{Trace}(tM)). \tag{1.29}$$

The proof for Eq (1.29) is given in Section 1.10 of [3]. Now recall that the special linear matrix Lie group is given by $SL(n) = \{A \in GL(n) \mid \det(A) = 1\}$. Let us define the matrix $A(t) = \exp(tM) \in SL(n)$ to be a smooth path in $SL(n)$, with $M \in \mathfrak{gl}(n)$ (for further explanation, please refer to Section 1.3.3). The condition $\det(A) = 1$ for $A \in SL(n)$ leads to,

$$\begin{aligned} 1 = \det(A(t)) &= \det(\exp(tM)) = \exp(\mathrm{Trace}(tM)) \\ \Rightarrow \mathrm{Trace}(M) &= 0. \end{aligned}$$

Since $\mathrm{Trace}[M, N] = 0$ for $M, N \in \mathfrak{sl}(n)$, the special linear matrix Lie algebra $\mathfrak{sl}(n)$ is hence given by $\mathfrak{sl}(n) = \{M \in \mathfrak{gl}(n) \mid \mathrm{Trace}(M) = 0\}$.

To find the condition for the **orthogonal matrix Lie algebra $\mathfrak{o}(n)$**, we again consider a smooth curve $A(t) = \exp(tM) \in O(n)$ at $t = 0$ through the unit element Id of the matrix orthogonal Lie group $O(n)$. Let matrix $M \in \mathfrak{gl}(n)$. Then by the orthogonality condition $A(t)^T A(t) = \mathrm{Id}$, we can expand this by Taylor series,

$$\mathrm{Id} = A(t)^T A(t) = (\mathrm{Id} + tM + \cdots)^T (\mathrm{Id} + tM + \cdots) = \mathrm{Id} + t(M + M^T) + O(t^2).$$

For matrices, recall that $\mathfrak{g} = T_I G$. Then we can find the condition for $\mathfrak{o}(n)$ by differentiating the above condition with respect to $t = 0$. This leads to the skew-symmetric matrix condition $M + M^T = 0$. Also, the commutator of two skew-symmetric matrices is a skew-symmetric matrix. Then, we can define the orthogonal matrix Lie algebra as, $\mathfrak{o}(n) = \{M \in \mathfrak{gl}(n) \mid M + M^T = 0\}$.

Note that for the special orthogonal matrix Lie group $SO(n)$, its corresponding matrix Lie algebra is also given by the group of skew-symmetric matrices as in $\mathfrak{o}(n)$, such that we have the **special orthogonal matrix Lie algebra**, $\mathfrak{so}(n) = \{M \in \mathfrak{gl}(n) \mid M + M^T = 0\}$ [3].

The steps to finding the condition for the **symplectic matrix Lie algebra $\mathfrak{sp}(2k)$** are similar to the discussion above. We want to consider a smooth curve $A(t) \in \mathfrak{sp}(2k)$ through the unit element of the group (at $t = 0$), such that $A(t) = \exp(tM) \in Sp(2k)$ for $M \in \mathfrak{gl}(2k)$. Then, expanding the condition for the symplectic matrix Lie group in Eq (1.22),

$$J = A(t)^T JA(t) = (\mathrm{Id} + tM^T + \cdots)J(\mathrm{Id} + tM + \cdots) = J + t(JM + M^T J) + O(t^2).$$

By the same argument above for the condition in $\mathfrak{o}(n)$, we find that the condition for the symplectic matrix Lie algebra is given by,

$$\mathfrak{sp}(n) = \{M \in \mathfrak{gl}(2k) \mid JM + M^T J = 0\}. \tag{1.30}$$

Furthermore, if we consider the matrix operator $M \in \mathfrak{sp}(2k)$ to be of the following block form, where A, B, C, D are $k \times k$ matrices,

$$M = \begin{bmatrix} A & B \\ C & D \end{bmatrix},$$

then, substituting this block form of M into Eq (1.30), we find,

$$\begin{aligned} 0 &= JM + M^T J \\ &= \begin{bmatrix} 0 & \mathrm{Id} \\ -\mathrm{Id} & 0 \end{bmatrix}\begin{bmatrix} A & B \\ C & D \end{bmatrix} + \begin{bmatrix} A^T & C^T \\ B^T & D^T \end{bmatrix}\begin{bmatrix} 0 & \mathrm{Id} \\ -\mathrm{Id} & 0 \end{bmatrix} \\ &= \begin{bmatrix} (C - C^T) & (D + A^T) \\ -(A + D^T) & (B^T - B) \end{bmatrix}. \end{aligned}$$

Hence the matrix $M \in \mathfrak{sp}(2k)$ has the block form,

$$M = \begin{bmatrix} A & B \\ C & -A^T \end{bmatrix}, \qquad \text{with } B = B^T,\ C = C^T.$$

1.3.3 Exponential Map and Left Lie Algebra Action

The link between Lie group G and its associated Lie algebra $\mathfrak{g}$ is the exponential map exp. It is defined as the map $\exp : \mathfrak{g} \to G$, such that for all matrices $v \in \mathfrak{g}$,

$$\exp(v) = \sum_{k=0}^{\infty} \frac{v^k}{k!}.$$

Note that the exponential map is a local diffeomorphism (smooth map with inverse) in the neighbourhood of $0 \in \mathfrak{g}$, by the inverse function theorem [30]. Also, given the nature of the exponential map, it maps $0 \in \mathfrak{g}$ to the identity element $e \in G$.

The exponential map allows for the computation of the flow for a given vector field (or infinitesimal generator) $v_{\mathcal{M}}$ for $v \in \mathfrak{g}$, such that the flow operator is,

$$\Psi_{v_{\mathcal{M}},t} = \exp(tv).$$

Using this definition, we can derive the Lie bracket for $u, v \in \mathfrak{g}$, in Eq (1.23) and Eq (1.24), which are now presented in the following lemma.

Lemma 1.2. *For all $u, v \in \mathfrak{g}$, we have the following definitions for the Lie bracket,*

$$\begin{aligned}[u,v] &= uv - vu, \\ [u,v] &= \frac{\partial^2}{\partial s \partial t}\rho(s)\sigma(t)\rho(-s)\bigg|_{s=t=0}.\end{aligned}$$

Proof. First, we compose the flow,

$$\begin{aligned}\Phi_{s,t}(y) &= \Psi_{u,s} \circ \Psi_{v,t} \circ \Psi_{u,-s} \circ \Psi_{v,-t}(y) \\ &= \exp(su) \circ \exp(tv) \circ \exp(-su) \circ \exp(-tv)(y).\end{aligned}$$

To write this in the usual Taylor series expansion for s and t at $s = t = 0$,

$$\begin{aligned}\Phi_{s,t}(y) &= \Phi_{0,0}(y) + s\left[\frac{\partial}{\partial s}\Phi_{s,0}(y)\bigg|_{s=0}\right] + t\left[\frac{\partial}{\partial t}\Phi_{0,t}(y)\bigg|_{t=0}\right] + \frac{s^2}{2!}\left[\frac{\partial^2}{\partial s^2}\Phi_{s,0}(y)\bigg|_{s=0}\right] + \\ &\frac{t^2}{2!}\left[\frac{\partial^2}{\partial t^2}\Phi_{0,t}(y)\bigg|_{t=0}\right] + \frac{2st}{2!}\left[\frac{\partial^2}{\partial s \partial t}\Phi_{s,t}(y)\bigg|_{s=t=0}\right] + O(s^3) + O(t^3) + O(st^2) + O(s^2t).\end{aligned}$$

But note that at $s = 0$, $\Phi_{0,t}(y) = e \circ \exp(tv) \circ e \circ \exp(-tv)(y) = y$, and at $t = 0$, $\Phi_{s,0}(y) = \exp(su) \circ e \circ \exp(-su) \circ e(y) = y$. So terms such as $\frac{\partial}{\partial s}\Phi_{s,0}(y)$, $\frac{\partial}{\partial t}\Phi_{0,t}(y)$ and other higher order terms based on those two all vanish. Also, the flow at $s = t = 0$ is the identity e. Hence, we are left with the following,

$$\Phi_{s,t}(y) = y + st\left[\frac{\partial^2}{\partial s \partial t}\Phi_{s,t}(y)\bigg|_{s=t=0}\right] + O(st^2) + O(s^2t).$$

Furthermore, if we label $H(y) = \frac{\partial^2}{\partial s \partial t}\Phi_{s,t}(y)\bigg|_{s=t=0}$, then we have,

$$\Phi_{s,t}(y) = y + stH(y) + O(st^2) + O(s^2t).$$

If u and v commute with each other, then the flow is simply $\Phi_{s,t}(y) = y$ and all subsequent terms are zero.

To compute $H(y)$, substitute the exponential expansion for the flow and differentiate,

$$\begin{aligned}H(y) &= \frac{\partial^2}{\partial s \partial t}\Phi_{s,t}(y)\bigg|_{s=t=0} \\ &= \frac{\partial}{\partial s}\left[\frac{\partial}{\partial t}\exp(su) \circ \exp(tv) \circ \exp(-su) \circ \exp(-tv)\bigg|_{t=0}\right]\bigg|_{s=0} \\ &= \frac{\partial}{\partial s}\left[\exp(su) \circ v \circ \exp(-su) \circ e + \exp(su) \circ e \circ \exp(-su) \circ (-v)\right]\bigg|_{s=0} \\ &= \frac{\partial}{\partial s}\left[\exp(su) \circ v \circ \exp(-su) - v\right]\bigg|_{s=0} \\ &= u \circ v \circ e + e \circ v \circ (-u) \\ &= u \circ v - v \circ u = [u,v].\end{aligned}$$

So, another way of writing the flow is,

$$\Phi_{s,t}(y) = y + st[u,v] + O(st^2) + O(s^2t),$$

where, for small s and t, the amount of commutivity between u and v is measured by the their commutator bracket $[u, v]$.

For Eq (1.24), which is the second definition in the Lemma, let there be two smooth curves $\rho : \mathbb{R} \to G$ and $\sigma : \mathbb{R} \to G$, such that $\rho(s) = \Psi_{u,s} = \exp(su)$ and $\sigma(t) = \Psi_{v,t} = \exp(tv)$. They must obey the conditions $\rho(0) = \sigma(0) = e$, $\rho'(0) = u$, and $\sigma'(0) = v$. Then, $H = [u, v]$ and the expression in Eq (1.24) are equivalent. This is because,

$$H(y) = \frac{\partial}{\partial s}\left[\frac{\partial}{\partial t}\rho(s) \circ \sigma(t) \circ \rho(-s)\bigg|_{t=0}\right]\bigg|_{s=0} = \frac{\partial}{\partial s}\left[\exp(su) \circ v \circ \exp(-su)\right]\bigg|_{s=0},$$

which has the same result as the previous definition using the flow $\Phi_{s,t}(y)$. □

Besides the definition of the Lie bracket, the exponential map also provides an important link between the Lie group action Λ and the lie algebra action λ discussed in Section 1.3.1 and Section 1.3.2. This relationship is given as the following in [28].

Definition 1.12. *If $\Lambda : G \times \mathcal{M} \to \mathcal{M}$ is a left Lie group action, then the left Lie algebra action $\lambda : \mathfrak{g} \times \mathcal{M} \to \mathcal{M}$ is given by,*

$$\lambda(v, p) = \Lambda(\exp(v), p), \tag{1.31}$$

for $v \in \mathfrak{g}$ close to 0, and $p \in \mathcal{M}$.

Given that we now have this connection between Lie group action and Lie algebra action in Definition 1.12, we can now rewrite the infinitesimal generator in Eq (1.26) as the following in Corollary 1.1 [11].

Corollary 1.1. *Let $\Lambda : G \times \mathcal{M} \to \mathcal{M}$ be the Lie Group Action, with Lie algebra $\mathfrak{g}$ on the manifold $\mathcal{M}$. Then the infinitesimal generator of the action corresponding to $v \in \mathfrak{g}$ is given by,*

$$v_{\mathcal{M}}(p) = \frac{d}{dt}\Lambda(\exp(tv), p)\big|_{t=0}, \qquad \forall p \in \mathcal{M}. \tag{1.32}$$

This infinitesimal generator is the vector field $F(p)$ on $\mathcal{M}$, given in Section 1.2.4.

This infinitesimal generator $v_{\mathcal{M}}$ describes the direction of the motion on $\mathcal{M}$, subject to the action of the Lie group element $\exp(tv)$. Note that this Lie group element at $t = 0$ is $\exp(0) = e$, and its derivative at $t = 0$ is simply given by $v \in \mathfrak{g}$. In turn, the Lie group action associated with its infinitesimal generator $v_{\mathcal{M}}$, is the flow of $v_{\mathcal{M}}$ on $\mathcal{M}$ [11].

Lemma 1.3. *Let $\Lambda : G \times \mathcal{M} \to \mathcal{M}$ be the Lie group action on $\mathcal{M}$. For all $v \in \mathfrak{g}$, $\Lambda(\exp(tv), \cdot) : \mathcal{M} \to \mathcal{M}$, is the flow of the infinitesimal generator $v_{\mathcal{M}}$.*

Proof. Recall that the flow must satisfy the definition of the integral curve in Definition 1.3. First, for the initial condition, $\Lambda(\exp(0), p) = \Lambda(e, p) = p$. Secondly, if we differentiate the Lie group action Λ with respect to t, and substitute a change of variable, we find,

$$\begin{aligned}
\tfrac{d}{dt}\Lambda(\exp(tv), p) &= \tfrac{d}{dt}\Lambda(\exp(sv), p)\big|_{s=t} \\
&= \tfrac{d}{ds}\Lambda(\exp((s+t)v), p)\big|_{s=0} \\
&= \tfrac{d}{ds}\Lambda(\exp(sv) \cdot \exp(tv), p)\big|_{s=0} \\
&= \tfrac{d}{ds}\Lambda(\exp(sv), \Lambda(\exp(tv), p))\big|_{s=0} && \text{by Definition 1.6} \\
&= v_{\mathcal{M}}(\Lambda(\exp(tv), p)) && \text{by Eq (1.32).}
\end{aligned}$$

□

1.3.4 The Differential of the Exponential Map and its Inverse

The definitions for both the differential of the exponential map dexp and its inverse dexp^{-1}, are important in the explanations of the Magnus Expansions, and Munthe-Kaas's RKMK Scheme in Lie group methods.

Before we begin however, we need to define the adjoint representation Ad_g of the Lie group G, and the linear adjoint operator ad which acts on the corresponding Lie algebra $\mathfrak{g}$ [19].

Definition 1.13. *The Adjoint representation of the Lie group,* $\mathrm{Ad}_g : \mathfrak{g} \to G$ *for all* $g \in G$*, is defined as,*

$$\mathrm{Ad}_g(v) = gvg^{-1}, \qquad \textit{for } g \in G, \textit{ and } v \in \mathfrak{g},$$

while the adjoint operator $\mathrm{ad}_u : \mathfrak{g} \to \mathfrak{g}$ *is the derivative of the Adjoint representation* Ad_g*, with respect to the element* $g \in G$ *at the identity* $e \in G$*, in the direction of* $u \in \mathfrak{g}$ *[24].*

Lemma 1.4. *The adjoint operator* $\mathrm{ad}_u : \mathfrak{g} \to \mathfrak{g}$ *is the Lie bracket introduced in Section 1.3.2.*

$$ad_u(v) = [u, v] = uv - vu, \qquad \forall u, v \in \mathfrak{g},$$

with the following recursive definition, $\mathrm{ad}_u^n(v) = \left[u, \mathrm{ad}_u^{n-1}(v)\right]$, $n = 1, 2, \ldots$.

Proof. Suppose g is an element in the matrix Lie group G, such that $g = \exp(tu)$, for all $u \in \mathfrak{g}$. We can differentiate the Adjoint representation $\mathrm{Ad}_g(v)$ at $v \in \mathfrak{g}$, with respect to $g = e \in G$, in the direction of $u \in \mathfrak{g}$,

$$\begin{aligned}
\left.\frac{d}{dg}\mathrm{Ad}_g(v)\right|_{g=e} &= \left.\frac{d}{dt}\mathrm{Ad}_{\exp(tv)}(v)\right|_{t=0} \\
&= \left.\frac{d}{dt}\exp(tu)v\exp(-tu)\right|_{t=0} \\
&= \left.\frac{d}{dt}\left(\mathrm{Id} + tu + \frac{(tu)^2}{2!} + \cdots\right) v \left(\mathrm{Id} - tu + \frac{(tu)^2}{2!} - \cdots\right)\right|_{t=0} \\
&= \left.(uv - vu + O(t))\right|_{t=0} \\
&= [u, v] .
\end{aligned}$$

□

In [28], we have the following relationships for the derivative of the exponential map $\exp(u + tv)$ for all $u, v \in \mathfrak{g}$, with respect to t, given by Theorem 1.3.

Theorem 1.3. *For* $u \in \mathfrak{g}$*, the derivative of the exponential map is given by* $\exp(u + t\cdot)' : \mathfrak{g} \to T_{\exp(u)}G$*, such that when applied to* $v \in \mathfrak{g}$*,*

$$\left.\frac{d}{dt}\exp(u + tv)\right|_{t=0} = \left.\exp(u + tv)'\right|_{t=0} = R'_{\exp(u)} \circ \mathrm{dexp}_u(v). \tag{1.33}$$

where $R'_{\exp(u)} : \mathfrak{g} \to T_{\exp(u)}G$*. The differential of the exponential map* $\mathrm{dexp}_u : \mathfrak{g} \to \mathfrak{g}$ *is the linear map, such that when applied to* $v \in \mathfrak{g}$*,*

$$\mathrm{dexp}_u(v) = \frac{\exp(\mathrm{ad}_u) - \mathrm{Id}}{\mathrm{ad}_u}(v) = \sum_{k=0}^{\infty} \frac{1}{(k+1)!}\mathrm{ad}_u^k(v), \tag{1.34}$$

and the inverse of dexp_u *is,*

$$\operatorname{dexp}_u^{-1}(v) = \frac{\mathrm{ad}_u}{\exp(\mathrm{ad}_u) - \mathrm{Id}}(v) = \sum_{k=0}^{\infty} \frac{B_k}{k!}\mathrm{ad}_u^k(v), \tag{1.35}$$

where B_k are the Bernoulli numbers, the first several of which are given by $B_0 = 1$, $B_1 = -1/2$, $B_2 = 1/6$, $B_4 = -1/30$, $B_6 = 1/42$, and $B_k = 0$ for all odd $k \geq 3$.

Proof. To see that the left and right hand sides of Eq (1.33) are equal, we can treat u and v as matrices, and work from both left and right hand side of Eq (1.33). If we expand the left hand side and differentiate with respect to $t = 0$, then we have,

$$\begin{aligned}
\frac{d}{dt}\exp(u+tv)\Big|_{t=0} &= \frac{d}{dt}\Big|_{t=0} \mathrm{Id} + (u+tv) + \frac{1}{2!}(u^2 + t(uv+vu) + O(t^2)) \\
&+\frac{1}{3!}(u^3 + t(u^2v + uvu + vu^2) + O(t^2)) + \cdots \\
&+\frac{1}{n!}(u^n + t(u^{n-1}v + u^{n-2}vu + \cdots + vu^{n-1}) + O(t^2)) + \cdots \\
&= v + \frac{1}{2!}(uv+vu) + \frac{1}{3!}(u^2v + uvu + vu^2) + \cdots \\
&+\frac{1}{n!}(u^{n-1}v + u^{n-2}vu + \cdots + u^{n-1-k}vu^k + \cdots + vu^{n-1}) + \cdots
\end{aligned} \tag{1.36}$$

Expanding the right hand side of Eq (1.33) we have,

$$\begin{aligned}
R'_{\exp(u)} \circ \operatorname{dexp}_u(v) &= \operatorname{dexp}_u(v)\cdot\exp(u) \\
&= \left(\frac{\exp(\mathrm{ad}_u) - \mathrm{Id}}{\mathrm{ad}_u}\right)(v)\cdot\exp(u) \\
&= \left(\mathrm{Id} + \frac{1}{2!}\mathrm{ad}_u + \frac{1}{3!}\mathrm{ad}_u^2 + \frac{1}{4!}\mathrm{ad}_u^3 + \cdots\right)(v)\cdot\exp(u) \\
&= \left(v + \frac{1}{2!}(uv - vu) + \frac{1}{3!}(u^2v - 2uvu + vu^2) + \cdots\right)\cdot\exp(u).
\end{aligned} \tag{1.37}$$

From here on, Prof. Butcher has devised a notation to help match the powers of terms of the left and right hand sides of Eq (1.33). Given that matrices may not commute, and the terms in the expansion generally involve some powers of matrix u multiplied to the left and/or right hand side of matrix v, we need to consider terms such as $u^{n-1-k}vu^k$. Let Z^n denotes a term with a combine total power n, for u and v matrices. Let X^k denotes the k-th power of u multiplied to v on the right hand side. For example,

$$\begin{aligned}
v &\mapsto Z^1X^0 = Z^1 \\
uv &\mapsto Z^2X^0 = Z^2 \\
vu &\mapsto Z^2X^1 \\
uvu &\mapsto Z^3X^1 \quad \text{etc.}
\end{aligned}$$

Hence, each term in the expansion can be represented uniquely by the Z and X notations. Now, Eq (1.36) becomes,

$$\begin{aligned}
\frac{d}{dt}\exp(u+tv)\Big|_{t=0} &\mapsto Z + \frac{1}{2!}Z^2(\mathrm{Id} + X) + \frac{1}{3!}Z^3(\mathrm{Id} + X + X^2) + \cdots \\
&+\frac{1}{n!}Z^n(\mathrm{Id} + X + \cdots + X^k + \cdots + X^{n-1}).
\end{aligned}$$

Similarly, Eq (1.37) can be rewritten with this notation. Note that $\exp(u)$ is multiplied to the right of $\mathrm{dexp}_u(v)$, so $\exp(u)$ can be rewritten as $\exp(ZX)$.

$$\begin{aligned}
R'_{\exp(u)}\mathrm{dexp}_u(v) &\mapsto \left(Z + \frac{1}{2!}Z^2(\mathrm{Id}-X) + \frac{1}{3!}Z^3(\mathrm{Id}-X)^2 + \cdots\right)\exp(ZX) \\
&= \left(\frac{\mathrm{Id} + Z(\mathrm{Id}-X) + \frac{1}{2!}Z^2(\mathrm{Id}-X)^2 + \cdots - \mathrm{Id}}{(\mathrm{Id}-X)}\right)\exp(ZX) \\
&= \left(\frac{\exp(Z(\mathrm{Id}-X)) - \mathrm{Id}}{(\mathrm{Id}-X)}\right)\exp(ZX) \\
&= \left(\frac{\exp(Z) - \exp(ZX)}{(\mathrm{Id}-X)}\right).
\end{aligned}$$

This expression can be expanded using the matrix exponential series, as,

$$\begin{aligned}
\left(\frac{\exp(Z) - \exp(ZX)}{(\mathrm{Id}-X)}\right) &= \frac{(\mathrm{Id} + Z + \frac{Z^2}{2!} + \frac{Z^3}{3!} + \cdots) - (\mathrm{Id} + ZX + \frac{(ZX)^2}{2!} + \frac{(ZX)^3}{3!} + \cdots)}{(\mathrm{Id}-X)} \\
&= \frac{Z(\mathrm{Id}-X) + \frac{Z^2(\mathrm{Id}-X^2)}{2!} + \frac{Z^3(\mathrm{Id}-X^3)}{3!} + \cdots}{(\mathrm{Id}-X)} \\
&= Z + \frac{1}{2!}Z^2(\mathrm{Id}+X) + \frac{1}{3!}Z^3(\mathrm{Id}+X+X^2) + \cdots
\end{aligned}$$

This is equivalent to the left hand side expression from $d/dt(\exp(u+tv)|_{t=0}$, which means Eq (1.36) and Eq (1.37) are the same.

To find the expression for $\mathrm{dexp}_u^{-1}(v)$, observe that dexp_u is an analytic function in ad_u, so we can invert the expression such that [19],

$$\begin{aligned}
\mathrm{dexp}_u^{-1}(v) &= \left(\frac{\mathrm{ad}_u}{\exp(\mathrm{ad}_u) - \mathrm{Id}}\right)(v) \\
&= \left(\mathrm{Id} - \frac{1}{2}\mathrm{ad}_u + \frac{1}{12}\mathrm{ad}_\mathrm{u}^2 - \frac{1}{720}\mathrm{ad}_\mathrm{u}^4 + \cdots\right)(v).
\end{aligned}$$

Note that the coefficients in the above series can be expressed by using the Bernoulli numbers B_k, so that the above in summation is Eq (1.35). □

In [19], the above notation for the differential of the exponential map in Theorem 1.3 is referred to as the **right trivialisation**: that is, the tangents at a point $Y \in G$ can be expressed as $AY \in \mathfrak{X}(\mathcal{M})$, where $A \in \mathfrak{g}$. If the differential equation is expressed using the **left trivialisation**, for tangents given as $YA \in \mathfrak{X}(\mathcal{M})$, then equations in Theorem 1.3 will all have slightly different expressions.

$$\begin{aligned}
\frac{d}{dt}\exp(u+tv)\bigg|_{t=0} &= L'_{\exp(u)} \circ \mathrm{dexp}_u(v) \\
\mathrm{dexp}_u(v)_L &= \sum_{k=0}^{\infty} \frac{(-1)^k}{(k+1)!}\mathrm{ad}_u^k(v), \\
\mathrm{dexp}_u^{-1}(v)_L &= v + \frac{1}{2}\mathrm{ad}_u(v) + \sum_{k=2}^{\infty}\frac{B_k}{k!}\mathrm{ad}_u^k(v).
\end{aligned}$$

Note that in this case, the Bernoulli numbers for $\mathrm{dexp}_u^{-1}(v)$ differ only in the sign for the second term $\mathrm{ad}_u^2(v)$ in the series.

1.3.5 Generic Presentation of Differential Equations on $\mathcal{M}$

The basic technique of defining differential equations on manifold $\mathcal{M}$ given by Definition 1.4, is by the movements of the Lie algebra action λ on $\mathcal{M}$. Let $f : \mathbb{R} \times \mathcal{M} \to \mathfrak{g}$, and $y : \mathbb{R} \to \mathcal{M}$ be the curve on $\mathcal{M}$, then by the right invariant vector field in Section 1.3.2, we have the differential equation, for $t \in \mathbb{R}$ and for all $p \in \mathcal{M}$,

$$y'(t) = R'_{y(t)}(f(t, y(t))) = X_{f(t,y(t))}(y(t)), \qquad y(0) = p. \tag{1.38}$$

In [28], Munthe-Kaas introduced the generic presentation of differential equations on $\mathcal{M}$, based on Lie algebra action $\lambda : \mathfrak{g} \times \mathcal{M} \to \mathcal{M}$.

Assumption 1.1. *We assume that there exists a Lie algebra $\mathfrak{g}$ with a Lie bracket $[\cdot,\cdot] : \mathfrak{g}\times\mathfrak{g} \to \mathfrak{g}$, a left Lie algebra action $\lambda : \mathfrak{g}\times\mathcal{M} \to \mathcal{M}$, and a function $f : \mathbb{R}\times\mathcal{M} \to \mathfrak{g}$, such that the differential equation for $y(t) \in \mathcal{M}$ can be written in the form:*

$$y'(t) = (\lambda_* f(t, y(t)))(y(t)), \qquad y(0) = p, \tag{1.39}$$

where $\lambda_* : \mathfrak{g} \to T\mathcal{M} \in \mathfrak{X}(\mathcal{M})$ is defined in Eq (1.26).

To see the relationship between the differential equation in Eq (1.38) and the generic form on $\mathcal{M}$ in Eq (1.39), in terms of **right trivialisation**, consider replacing the right multiplication in Lie group,

$$R_p(g) = g \cdot p, \quad \forall g, p \in G,$$

with the Lie group action, when $G = \mathcal{M}$ [29],

$$\Lambda(g, p) = \Lambda_p(g) = g \cdot p, \quad g \in G,\ p \in \mathcal{M}.$$

Given that $g = \exp(tv) \in G$ for $f : \mathbb{R} \times \mathcal{M} \to \mathfrak{g}$, then Eq (1.38) becomes,

$$\begin{aligned} y' &= R'_y(f(t, y)) = T_e R_y(f(t, y)) \\ &= \frac{d}{dt}\Lambda(\exp(tf(t, y)), y)\bigg|_{t=0} = \frac{d}{dt}\lambda(tf(t, y), y)\bigg|_{t=0} \\ &= \lambda_*(f(t, y))(y). \end{aligned}$$

In other words, let us say that the original differential equation is given by $\dot{y}(t) = F(y(t))$. Then, one of the major assumptions behind the theory for the RKMK method is that there exists a Lie group action Λ such that for every point $y(t) \in \mathcal{M}$, $\forall t \in \mathbb{R}$, we can rewrite the original differential equation in the form,

$$F(y(t)) = \frac{d}{dt}\bigg|_{t=0} \Lambda(\exp(tf(t, y)), y), \qquad \text{where } f : \mathbb{R} \times \mathcal{M} \to \mathfrak{g}.$$

Once again, this defines the infinitesimal generator $v_{\mathcal{M}}$ of the action Λ, corresponding to the function $f : \mathbb{R} \times \mathcal{M} \to \mathfrak{g}$.

Furthermore, the generic form of Eq (1.39) for matrices can be rewritten in the more familiar form [19], using the differential of the exponential given in Theorem 1.3,

$$\begin{aligned} y'(t) &= \frac{d}{dt}\Lambda(\exp(tf(t, y)), y)\bigg|_{t=0} = \frac{d}{dt}(\exp(tf(t, y)) \cdot y)\bigg|_{t=0} \\ &= [\operatorname{dexp}_0(f(t, y)) \cdot \exp(0)] \cdot y \\ &= f(t, y) \cdot y, \qquad y(0) = p. \end{aligned}$$

In this case, the flow of the vector field $\lambda_* f(t, y) \in \mathfrak{X}(\mathcal{M})$ is given by the Lie group action,

$$y(t) = \Lambda(\exp(tf(t, y)), p) = \exp(tf(t, y)) \cdot p.$$

1.3.6 Lie Type Equations

A special case of Eq (1.39) is when the function $f : \mathbb{R} \times \mathcal{M} \to \mathfrak{g}$ depends only on $t \in \mathbb{R}$, such that we have the function $v : \mathbb{R} \to \mathfrak{g}$ to replace f. These are called the **(linear) Lie type equations** [19].

Using the right invariant vector field, Eq (1.39) becomes,

$$y'(t) = (\lambda_* v(t))(y(t)) = R'_{y(t)}(v(t)) = X_{v(t)}(y(t)), \qquad y(0) = p. \tag{1.40}$$

where, as $t \in \mathbb{R}$ varies, then $y : \mathbb{R} \to \mathcal{M}$ and $v : \mathbb{R} \to \mathfrak{g}$ are two curves on $G = \mathcal{M}$ and $\mathfrak{g}$ respectively. X is another notation for the right invariant vector field. Note that the same is true for the left invariant vector field, denoted $L'_{y(t)}(v(t))$, as shown in [27].

Now, let $s : \mathbb{R} \to \mathcal{M}$ be the integral curve of Eq (1.40) starting at $s(0) = e \in G = \mathcal{M}$. Then, s is the fundamental solution which satisfies,

$$s'(t) = R'_{s(t)}(v(t)) = X_{v(t)}(s(t)), \qquad s(0) = e. \tag{1.41}$$

Then, the **general solution** $y(t) \in \mathcal{M}$ of Eq (1.40) with respect to the right invariant vector field $X_{v(t)}$, is now given by the 'left invariant' flow Ψ such that,

$$y(t) = \Psi_{X_{v(t)},t}(p) = L_{s(t)}(p) = s(t) \cdot p. \tag{1.42}$$

The above is presented more formally in [27] as the following.

Lemma 1.5. *Let there be an arbitrary curve $y : \mathbb{R} \to G = \mathcal{M}$, which satisfies the Lie type equation in Eq (1.40). Then if the curve $v : \mathbb{R} \to \mathfrak{g}$ is given by,*

$$v(t) = R'_{y(t)^{-1}}(y'(t)),$$

for all $t \in \mathbb{R}$, the general solution y to Eq (1.40) is given by Eq (1.42), and the fundamental solution s must satisfy Eq (1.41).

1.3.7 The Baker-Campbell-Hausdorff Formula

For two arbitrary matrices A and B which are usually non-commutating, we want to find a matrix $C(t)$ such that for small $t \in \mathbb{R}$,

$$\exp(tA)\exp(tB) = \exp(C(t)), \tag{1.43}$$

where $C(t)$ is the Taylor expansion, $C(t) = \sum\limits_{i=1}^{\infty} C_i t^i$. The goal is to find explicit expressions for the coefficients in $C(t)$ in terms of commutators acting on matrices A and B. The expression for $C(t)$ is known as the Baker-Campbell-Hausdorff (BCH) Formula.

In [14], the following Lemma provides a differential equation for $C(t)$ in Eq(1.44).

Lemma 1.6. *Let A and B be non-commuting matrices. Then, Eq (1.43) is true, and C is the solution to the initial value problem,*

$$\dot{C} = A + B + \frac{1}{2}[A - B, C] + \sum_{k=2}^{\infty} \frac{B_k}{k!} \mathrm{ad}_C^k (A + B), \quad C(0) = 0. \tag{1.44}$$

where B_k denote the Bernoulli numbers.

Proof. For small s and t, and a smooth matrix function $Z(s,t)$, consider the following expression,

$$\exp(sA)\exp(tB) = \exp(Z(s,t)). \tag{1.45}$$

Differentiating the left and right hand side of Eq (1.45) with respect to s using the right trivialisation in Theorem 1.3, and substitute Eq (1.45) into the expression, we find,

$$\begin{aligned} A\exp(sA)\exp(tB) &= \frac{\partial}{\partial s}\exp(Z(s,t)) \\ &= \operatorname{dexp}_{Z(s,t)}\left(\frac{\partial}{\partial s}Z(s,t)\right)\cdot\exp(Z(s,t)) \\ &= \operatorname{dexp}_{Z(s,t)}\left(\frac{\partial}{\partial s}Z(s,t)\right)\cdot(\exp(sA)\exp(tB)) \\ \Rightarrow A &= \operatorname{dexp}_{Z(s,t)}\left(\frac{\partial}{\partial s}Z(s,t)\right). \end{aligned}$$

To find an expression for $d/dsZ(s,t)$, we take the inverse of dexp,

$$\begin{aligned} \frac{\partial}{\partial s}Z(s,t) &= \operatorname{dexp}^{-1}_{Z(s,t)}(A) = \sum_{k=0}^{\infty}\frac{B_k}{k!}\operatorname{ad}^k_{Z(s,t)}(A) \\ &= A - \frac{1}{2}\operatorname{ad}_{Z(s,t)}(A) + \sum_{k=2}^{\infty}\frac{B_k}{k!}\operatorname{ad}^k_{Z(s,t)}(A). \end{aligned} \tag{1.46}$$

Next, if we take the inverse of Eq (1.45),

$$\exp(-sA)\exp(-tB) = \exp(-Z(s,t)),$$

and differentiate both sides with respect to t, we can find an expression for the partial derivative of $Z(s,t)$ with respect to t.

$$\begin{aligned} \frac{\partial}{\partial t}Z(s,t) &= \operatorname{dexp}^{-1}_{-Z(s,t)}(B) = \sum_{k=0}^{\infty}\frac{B_k}{k!}\operatorname{ad}^k_{-Z(s,t)}(B) \\ &= B - \frac{1}{2}\operatorname{ad}_{-Z(s,t)}(B) + \sum_{k=2}^{\infty}\frac{B_k}{k!}\operatorname{ad}^k_{-Z(s,t)}(B) \\ &= B + \frac{1}{2}\operatorname{ad}_{Z(s,t)}(B) + \sum_{k=2}^{\infty}\frac{B_k}{k!}\operatorname{ad}^k_{Z(s,t)}(B), \end{aligned} \tag{1.47}$$

which is just the left trivialisation of $\operatorname{dexp}^{-1}_{Z(s,t)}(B)_L$. If we compare Eq (1.43) with Eq (1.45), we see that $C(t) = Z(s,t)|_{s=t}$. Then $C(t)$ is the solution to the differential equation,

$$\frac{d}{dt}C(t) = \frac{\partial}{\partial s}Z(s,t)\bigg|_{s=t} + \frac{d}{dt}Z(s,t)\bigg|_{t}.$$

Substituting Eq (1.46) and Eq (1.47), we have,

$$\begin{aligned} \frac{d}{dt}C(t) &= \left(A - \frac{1}{2}\operatorname{ad}_{Z(s,t)}(A) + \sum_{k=2}^{\infty}\frac{B_k}{k!}\operatorname{ad}^k_{Z(s,t)}(A)\right)\bigg|_{s=t} \\ &\quad + \left(B + \frac{1}{2}\operatorname{ad}_{Z(s,t)}(B) + \sum_{k=2}^{\infty}\frac{B_k}{k!}\operatorname{ad}^k_{Z(s,t)}(B)\right)\bigg|_{t} \\ &= A + B + \frac{1}{2}[A+B, C(t)] + \sum_{k=2}^{\infty}\frac{B_k}{k!}\operatorname{ad}^k_{C(t)}(A+B). \end{aligned}$$

This completes the proof to Lemma 1.6. □

Let $C(t)$ be given by a Taylor series $C(t) = \sum_{i=1}^{\infty} C_i t^i$. Differentiating with respect to t yields,

$$\dot{C}(t) = C_1 + 2C_2 t + 3C_3 t^2 + \cdots \tag{1.48}$$

If we substitute the same Taylor series $C(t) = \sum C_i t^i$ into Eq (1.44), and match the coefficients C_i between this and Eq (1.48), we find the expressions for C_i to be,

$$\begin{aligned}
C_1 &= A + B, \\
C_2 &= \frac{1}{2}[A, B], \\
C_3 &= \frac{1}{12}[A, [A, B]] + \frac{1}{12}[B, [B, A]], \\
C_4 &= \frac{1}{24}[A, [B, [B, A]]], \\
\ldots
\end{aligned}$$

Where expressions from C_4 onward are simplified using the Jacobi Identity in Section 1.3.2. Hence, the matrix $C(t)$ has the expression,

$$\begin{aligned}
C(t) &= C_1 t + C_2 t^2 + C_3 t^3 + O(t^4) \\
&= (A+B)t + \frac{1}{2}[A, B]t^2 + \frac{1}{12}([A, [A, B]] + [B, [B, A]])t^3 + O(t^4),
\end{aligned}$$

which is known as the BCH formula.

1.3.8 Left and Right Trivialisations

In the previous sections, most of the definitions given are based on the right trivialisations, for differential equations with tangents given by $AY \in \mathfrak{X}(\mathcal{M})$, where $A \in \mathfrak{g}$ and $Y \in G$. In most expressions, there are slight differences between the left and right conventions. One important distinction between the two is that the left convention denotes calculations done in the space coordinates of a moving object, while the right convention denotes that of the body coordinates of the object. The space coordinate is a frame fixed in space, while the body coordinate is a frame fixed within the body, so that this frame is moving with the body, observed with respect to the space coordinates.

For easy reference, I have listed some of the more frequently used calculations in both their left and right hand trivialisations for matrices, in Table 1.1.

Table 1.1: Table of expressions using left and right trivialisations.

Left Trivialisation	Right Trivialisation
$L_a(b) = a \cdot b$ $L : G \times G \to G$	$R_a(b) = b \cdot a$ $R : G \times G \to G$
$L'_p(v) = p \cdot v$ $L' : \mathfrak{g} \to T_pG$	$R'_p(v) = v \cdot p$ $R' : \mathfrak{g} \to T_pG$
$\lambda(p, v) = R_{\exp(v)}(p) = p \cdot \exp(v)$ $\lambda : \mathcal{M} \times \mathfrak{g} \to \mathcal{M}$	$\lambda(v, p) = L_{\exp(v)}(p) = \exp(v) \cdot p$ $\lambda : \mathfrak{g} \times \mathcal{M} \to \mathcal{M}$
$(\lambda_* v)(p) = \frac{d}{dt}\lambda(p, tv)\big\|_{t=0}$ $\lambda_* : \mathfrak{g} \to \mathfrak{X}(\mathcal{M})$	$(\lambda_* v)(p) = \frac{d}{dt}\lambda(tv, p)\big\|_{t=0}$ $\lambda_* : \mathfrak{g} \to \mathfrak{X}(\mathcal{M})$
$\frac{d}{dt}\exp(u + tv)\big\|_{t=0} = L'_{\exp(u)} \circ \mathrm{dexp}_u(v)$	$\frac{d}{dt}\exp(u + tv)\big\|_{t=0} = R'_{\exp(u)} \circ \mathrm{dexp}_u(v)$
$\mathrm{dexp}_u(v)_L = \sum\limits_{k=0}^{\infty} \frac{(-1)^k}{(k+1)!}\mathrm{ad}_u^k(v)$ $\mathrm{dexp} : \mathfrak{g} \times \mathfrak{g} \to \mathfrak{g}$	$\mathrm{dexp}_u(v) = \sum\limits_{k=0}^{\infty} \frac{1}{(k+1)!}\mathrm{ad}_u^k(v)$ $\mathfrak{g} \times \mathfrak{g} \to \mathfrak{g}$
$\mathrm{dexp}_u^{-1}(v)_L = v + \frac{1}{2}\mathrm{ad}_u(v) + \sum\limits_{k=2}^{\infty} \frac{B_k}{k!}\mathrm{ad}_u^k(v)$	$\mathrm{dexp}_u^{-1}(v) = v - \frac{1}{2}\mathrm{ad}_u(v) + \sum\limits_{k=2}^{\infty} \frac{B_k}{k!}\mathrm{ad}_u^k(v)$

CHAPTER 2

Magnus Series Expansion

In Chapter 2, we study the approach of the Magnus series expansion, when solving the matrix differential equation,

$$\dot{y}(t) = A(t)y(t),$$

using Picard iterations. In the case when $y(t)$ is an element in a Lie group, then the matrix $A(t)$ is in the corresponding Lie algebra, and the differential equation becomes a Lie group equation. So, the Magnus series expansion must also respect the structure of the Lie group, if it is to be used as the basis for a geometric integrator.

Also, we introduce the rooted tree theory associated with the terms in the Magnus series expansion. Each term in the expansion features a number of nested commutators and multiple integrals from the fixed point iteration scheme, and they become increasingly complex as we expand to higher orders. We will see that using the rooted trees provides a 'short-hand' method of finding all the recursive terms required for a particular order in the Magnus series expansion.

Also, we will discuss briefly the idea of a power m associated with the rooted trees from the Magnus series expansion, and the concept that a truncated Magnus series respects the time-symmetry property. This leads to major reduction in the number of terms required in the expansion up to a particular order. Hence, it significantly reduces the computational cost of the Magnus series expansion.

Due to the nature of the multiple integrals in the terms of the expansion, we need to understand how to use multivariate quadrature to approximate such multi-dimensional integrals, and in particular, the important role that Legendre polynomial of degree s plays in the quadrature of a $2s$ order approximation. In the last section, the subject of quadrature and some implementation issues will be considered.

2.1 Magnus Series Expansions

In Section 1.3.5, we were introduced to the differential equation on the matrix Lie group, which has the form,

$$\dot{y}(t) = f(t, y)y(t), \qquad y(0) = p.$$

Note that if the function $f : \mathbb{R} \times \mathcal{M} \rightarrow \mathfrak{g}$ depends only on t, such that we have the function, say $A : \mathbb{R} \rightarrow \mathfrak{g}$, then the above becomes a linear matrix differential equation,

$$\dot{y}(t) = A(t)y(t), \qquad y(0) = y_0. \tag{2.1}$$

In this case, the solution to Eq (2.1) can be found by the Magnus series expansion, and we will follow the discussions in [14] and [19] in the section below. Note that Eq (2.1) is a Lie group equation if $y(t)$ is an element in a matrix Lie group G, and the matrix $A(t)$ is in the corresponding Lie algebra $\mathfrak{g}$. So, the Magnus series expansion also needs to respect the structure of the group under consideration.

The basic idea is that in the general non-commutative case, we want to search for a matrix function $\Omega(t)$ such that the solution $y(t)$ for Eq (2.1) is described as,

$$y(t) = \exp(\Omega(t))y_0. \tag{2.2}$$

It turns out that to find the explicit expression for $\Omega(t)$, we need to solve another differential equation [14],

$$\dot{\Omega}(t) = \operatorname{dexp}^{-1}_{\Omega(t)}(A(t)), \qquad \Omega(0) = 0. \tag{2.3}$$

To derive Eq (2.3), let us assume Eq (2.2) holds and we differentiate it with respect to t using Theorem 1.3 for the derivative of the exponential map.

$$\begin{aligned}
\frac{d}{dt}y(t) &= \frac{d}{dt}\left(\exp(\Omega(t))y_0\right) \\
\dot{y}(t) &= R'_{\exp(\Omega(t))} \circ \operatorname{dexp}_{\Omega(t)}\left(\frac{d}{dt}\Omega(t)\right) \cdot y_0 \\
&= \operatorname{dexp}_{\Omega(t)}\left(\dot{\Omega}(t)\right) \cdot \exp(\Omega(t)) \cdot y_0.
\end{aligned} \tag{2.4}$$

Comparing Eq (2.4) with Eq (2.1) and Eq (2.2), we find that,

$$A(t) = \operatorname{dexp}_{\Omega(t)}\left(\dot{\Omega}(t)\right).$$

Inverting the above yields the desired relationship in Eq (2.3).

Recall that the series for dexp^{-1} has the form given by Eq (1.35) in Theorem 1.3. Then, Eq (2.3) can be further expanded as,

$$\dot{\Omega}(t) = A(t) - \frac{1}{2}[\Omega(t), A(t)] + \frac{1}{12}[\Omega(t), [\Omega(t), A(t)]] + \cdots \tag{2.5}$$

Applying the idea of **Picard iterations** [2], we can find an explicit approximation to $\Omega(t)$. This approximation is called the Magnus series expansion [14], and is derived according to the steps below.

Let Ω_0 be the initial estimate to the solution $\Omega(t)$. Note that Eq (2.3) has the initial condition $\Omega(0) = 0$, so we can let our initial estimate be $\Omega_0 = 0$. Now, if we substitute $\Omega_0 = 0$ into Eq (2.5), we are left with,

$$\dot{\Omega}(t) = A(t), \qquad \Omega(0) = \Omega_0 = 0,$$

and integrate with respect to t,

$$\int_0^t \dot{\Omega}(\xi)d\xi = \int_0^t A(\xi)d\xi.$$

So, the first iteration, denoted by $\Omega_1(t)$ yields the following approximation to $\Omega(t)$,

$$\begin{aligned}\Omega(t) - \Omega(0) &= \int_0^t A(\xi)d\xi \\ \Rightarrow \Omega_1(t) &= \int_0^t A(\xi)d\xi + \Omega_0 = \int_0^t A(\xi)d\xi. \end{aligned} \tag{2.6}$$

Now, we substitute Eq (2.6) into Eq (2.5),

$$\dot{\Omega}(t) = A(t) - \frac{1}{2}\left[\Omega_1(t), A(t)\right] + \frac{1}{12}\left[\Omega_1(t), \left[\Omega_1(t), A(t)\right]\right] - \cdots,$$

and integrate to find the second iterate $\Omega_2(t)$,

$$\Omega_2(t) = \int_0^t A(\xi)d\xi - \frac{1}{2}\int_0^t \left[\Omega_1(\xi), A(\xi)\right] d\xi + \frac{1}{12}\int_0^t \left[\Omega_1(\xi), \left[\Omega_1(\xi), A(\xi)\right]\right] d\xi + \cdots \tag{2.7}$$

Similarly, we substitute Eq (2.7) into Eq (2.5) to find the third iterate $\Omega_3(t)$,

$$\Omega_3(t) = \int_0^t A(\xi)d\xi - \frac{1}{2}\int_0^t \left[\Omega_2(\xi), A(\xi)\right] d\xi + \frac{1}{12}\int_0^t \left[\Omega_2(\xi), \left[\Omega_2(\xi), A(\xi)\right]\right] d\xi + \cdots \tag{2.8}$$

Back substituting $\Omega_1(t)$ and $\Omega_2(t)$ into Eq (2.8) for $\Omega_3(t)$ yields a complex expression involving multiple integrals inside nested commutators. After some expansion, the first few terms are displayed below (for all terms up to the triple integrals),

$$\begin{aligned}\Omega_3(t) = & \int_0^t A(\xi_1)d\xi_1 - \frac{1}{2}\int_0^t \left[\int_0^{\xi_1} A(\xi_2)d\xi_2, A(\xi_1)\right] d\xi_1 \\ & + \frac{1}{4}\int_0^t \left[\int_0^{\xi_1}\left[\int_0^{\xi_2} A(\xi_3)d\xi_3, A(\xi_2)\right] d\xi_2, A(\xi_1)\right] d\xi_1 \\ & + \frac{1}{12}\int_0^t \left[\int_0^{\xi_1} A(\xi_2)d\xi_2, \left[\int_0^{\xi_1} A(\xi_3)d\xi_3, A(\xi_1)\right]\right] d\xi_1 \\ & - \frac{1}{24}\int_0^t \left[\int_0^{\xi_1}\left[\int_0^{\xi_2} A(\xi_3)d\xi_3, \left[\int_0^{\xi_2} A(\xi_3)d\xi_3, A(\xi_2)\right]\right] d\xi_2, A(\xi_1)\right] d\xi_1 \\ & - \frac{1}{24}\int_0^t \left[\int_0^{\xi_1} A(\xi_2)d\xi_2, \left[\int_0^{\xi_1}\left[\int_0^{\xi_2} A(\xi_3)d\xi_3, A(\xi_2)\right] d\xi_2, A(\xi_1)\right]\right] d\xi_1 \\ & - \frac{1}{24}\int_0^t \left[\int_0^{\xi_1}\left[\int_0^{\xi_2} A(\xi_3)d\xi_3, A(\xi_2)\right] d\xi_2, \left[\int_0^{\xi_1} A(\xi_2)d\xi_2, A(\xi_1)\right]\right] d\xi_1 \\ & - \frac{1}{8}\int_0^t \left[\int_0^{\xi_1}\left[\int_0^{\xi_2}\left[\int_0^{\xi_3} A(\xi_4)d\xi_4, A(\xi_3)\right] d\xi_3, A(\xi_2)\right] d\xi_2\right] d\xi_1 + \cdots\end{aligned}$$

Hence, the Magnus series expansion can be rewritten as a summation of terms $H_k(t)$, where each H_k is a linear combination of terms that include exactly $k+1$ integrals (and k commutators) [19],

$$\Omega(t) = \sum_{k=0}^{\infty} H_k(t), \tag{2.9}$$

where the first few terms in Eq (2.9) are given by,

$$\begin{aligned}
H_0(t) &= \int_0^t A(\xi_1)d\xi_1 \\
H_1(t) &= -\frac{1}{2}\int_0^t \left[\int_0^{\xi_1} A(\xi_2)d\xi_2, A(\xi_1)\right] d\xi_1 \\
H_2(t) &= \frac{1}{4}\int_0^t \left[\int_0^{\xi_1}\left[\int_0^{\xi_2} A(\xi_3)d\xi_3, A(\xi_2)\right] d\xi_2, A(\xi_1)\right] d\xi_1 \\
&\quad +\frac{1}{12}\int_0^t \left[\int_0^{\xi_1} A(\xi_2)d\xi_2, \left[\int_0^{\xi_1} A(\xi_3)d\xi_3, A(\xi_1)\right]\right] d\xi_1 \\
H_3(t) &= -\frac{1}{24}\int_0^t \left[\int_0^{\xi_1}\left[\int_0^{\xi_2} A(\xi_3)d\xi_3, \left[\int_0^{\xi_2} A(\xi_3)d\xi_3, A(\xi_2)\right]\right] d\xi_2, A(\xi_1)\right] d\xi_1 \\
&\quad -\frac{1}{24}\int_0^t \left[\int_0^{\xi_1} A(\xi_2)d\xi_2, \left[\int_0^{\xi_1}\left[\int_0^{\xi_2} A(\xi_3)d\xi_3, A(\xi_2)\right] d\xi_2, A(\xi_1)\right]\right] d\xi_1 \\
&\quad -\frac{1}{24}\int_0^t \left[\int_0^{\xi_1}\left[\int_0^{\xi_2} A(\xi_3)d\xi_3, A(\xi_2)\right] d\xi_2, \left[\int_0^{\xi_1} A(\xi_2)d\xi_2, A(\xi_1)\right]\right] d\xi_1 \\
&\quad -\frac{1}{8}\int_0^t \left[\int_0^{\xi_1}\left[\int_0^{\xi_2}\left[\int_0^{\xi_3} A(\xi_4)d\xi_4, A(\xi_3)\right] d\xi_3, A(\xi_2)\right] d\xi_2\right] d\xi_1
\end{aligned}$$

Note that Eq (2.9) is the approximate solution to the differential equation Eq (2.3). It is obvious that the terms in the Magnus series expansion are becoming increasingly complex for terms involving high values of k. It is also observed that each term is made out of integrals and commutators on the matrix function $A(t)$ in Eq (2.3), and a shorthand notation using rooted binary trees has been extensively developed to represent each terms in the Magnus expansion. In the following section, we will present a brief summary of the construction of such rooted trees.

2.2 Rooted Trees

We follow the discussion in [19] and [20] for the theory of the rooted trees for the Magnus series expansion.

First, recall that each term in the Magnus series expansion can be grouped together by k commutators or $(k+1)$ integrals. If we, for the moment, ignore the 'outer-most' integral from 0 to t, then each term in the expansion has equal numbers of commutators and integrals. We will denote such terms by $C_\tau(\xi)$, their scalar constants $\alpha(\tau)$, and the binary tree $\tau \in \mathbb{T}_k$ associated with each term in the expansion. Note that the set $\mathbb{T}_k$ includes all trees with the same number of k integrals and k commutators. Hence, we can rewrite the approximation in Eq (2.9) as the following,

$$\Omega(t) = \sum_{k=0}^{\infty} H_k(t) = \sum_{k=0}^{\infty} \sum_{\tau \in \mathbb{T}_k} \alpha(\tau) \int_0^t C_\tau(\xi) d\xi, \qquad t \geq 0. \tag{2.10}$$

To define $C_\tau(\xi)$, we start by defining $C_\tau(\xi)$ at $k = 0$,

$$C_{\tau_0}(\xi) = A(\xi), \qquad \text{where } \tau_0 \in \mathbb{T}_0 \text{ for } k = 0,$$

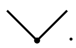
and let $\mathbb{T}_0$ be the set of rooted trees with one single vertex, that is,

$$A(\xi) \quad \Rightarrow \quad \bullet \ \in \mathbb{T}_0.$$

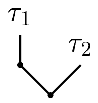
From Picard iteration in Section 2.1, for $k \geq 1$ and $\tau \in \mathbb{T}_{m_1+m_2+1}$, we can define $C_\tau(\xi)$ to be,

$$C_\tau(\xi) = \left[\int_0^\xi C_{\tau_1}(\xi_1) d\xi_1, C_{\tau_2}(\xi) \right], \tag{2.11}$$

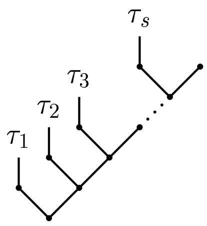
where the rooted trees $\tau_1 \in \mathbb{T}_{m_1}$ and $\tau_2 \in \mathbb{T}_{m_2}$. Pictorially, we denote an integral as a vertical branch, and a commutator as two branches joined by a common root, that is,

 and .

Then, Eq (2.11) is associated with the rooted trees $\tau \in \mathbb{T}_{m_1+m_2+1}$ in the following way,

$$\left[\int_0^\xi C_{\tau_1}(\xi_1) d\xi_1, C_{\tau_2}(\xi) \right] \quad \Rightarrow \quad \tau = $$  $$\in \mathbb{T}_{m_1+m_2+1}. \tag{2.12}$$

By using Eq (2.12), all binary trees associated with all the terms in the Magnus series expansions can be recursively defined. Also, the constant $\alpha(\tau)$ can be defined by the rooted tree theory. Generally, a binary rooted tree τ representing the terms in the Magnus series expansion has the form, for $s \geq 1$,

$$\tau = $$  (2.13)

Note that the sub-trees τ_i for $i \geq 2$ are defined slightly differently in Eq (2.13) than those in Eq (2.12). With Eq (2.13), the constant $\alpha(\tau)$ is calculated by [19],

$$\begin{aligned} \alpha(\tau_0) &= \alpha(\ \bullet\ ) = 1 \\ \alpha(\tau) &= \frac{B_s}{s!} \prod_{i=1}^{s} \alpha(\tau_i), \qquad s \in \mathbb{Z}_+, \end{aligned} \tag{2.14}$$

where B_s are the Bernoulli numbers. Hence, with the help of the rooted trees, we can now continue to derive the higher order terms in the Magnus series expansion in Eq (2.10). Furthermore, for smooth matrix function $A(t)$, it can be proved that for $k \in \mathbb{Z}_+$ [19],

$$\int_0^t C_\tau(\xi)d\xi = O(t^{k+1}) \quad \Rightarrow \quad \tau \in \mathbb{T}_k. \tag{2.15}$$

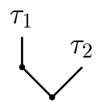
We will see in the next section that this order can be improved upon, and that a lot less terms are actually needed than stated in Eq (2.15), if the method is time-symmetric.

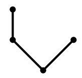
For now, we will look in detail at how the trees are built using the rooted tree theory outlined previously. In the following, we will derive all of the rooted trees τ in the set $\mathbb{T}_k = \mathbb{T}_{m_1+m_2+1}$ for $0 \leq k \leq 3$.

For $k = 0$, we defined the rooted tree to be,

$$\tau = \quad \bullet \quad \in \mathbb{T}_0.$$

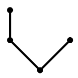
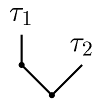
For $k = 1$, we have $m_1 = m_2 = 0$, so the rooted tree $\tau \in \mathbb{T}_1$ is given by,

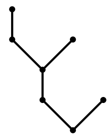
$\tau_1 \;=\; \bullet \qquad \tau_2 \;=\; \bullet \qquad \tau \;=$ 

$=$  $\in \mathbb{T}_1$

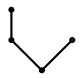
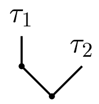
For $k = 2$, we have two choices for the pair m_1 and m_2,

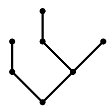
Either $m_1 = 1$ and $m_2 = 0$,

$\tau_1 \;=$  $\qquad \tau_2 \;=\; \bullet \qquad \tau \;=$ 

$=$  $\in \mathbb{T}_2$

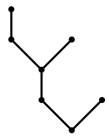
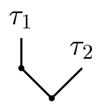
Or $m_1 = 0$ and $m_2 = 1$,

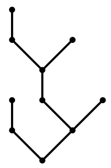
$\tau_1 \;=\; \bullet \qquad \tau_2 \;=$  $\qquad \tau \;=$ 

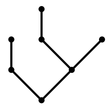
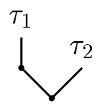
$=$  $\in \mathbb{T}_2$

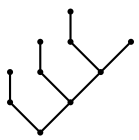
For $k = 3$, we have the following choices of m_1 and m_2,

Either $m_1 = 0$, and $m_2 = 2$, in which case we have two possibilities,

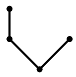
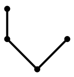
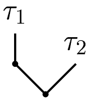
$\tau_1 \;=\; \bullet \qquad \tau_2 \;=$  $\qquad \tau \;=$ 

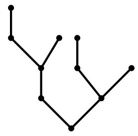
$=$  $\in \mathbb{T}_3$

$\tau_1 \;=\; \bullet \qquad \tau_2 \;=$  $\qquad \tau \;=$ 

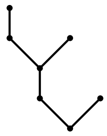
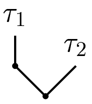
$=$  $\in \mathbb{T}_3$

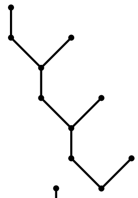
Or $m_1 = 1$ and $m_2 = 1$,

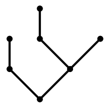
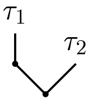
$\tau_1 =$  $\tau_2 =$  $\tau =$ 

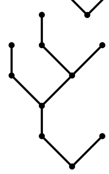
$=$  $\in \mathbb{T}_3$

Or $m_1 = 2$ and $m_2 = 0$,

$\tau_1 =$  $\tau_2 = \bullet$ $\tau =$ 

$=$  $\in \mathbb{T}_3$

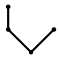
$\tau_1 =$  $\tau_2 = \bullet$ $\tau =$ 

$=$  $\in \mathbb{T}_3$

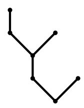
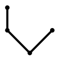
For each tree τ, we can now find the constant $\alpha(\tau)$ by the tree structure shown in Eq (2.13) and the formula in Eq (2.14). Recall that,

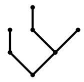
$$\alpha(\ \bullet\ ) = 1.$$

Then, for $\mathbb{T}_1$ we find,

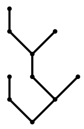
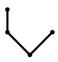
$\tau =$  $\tau_1 = \bullet$ $\alpha(\tau) = \frac{B_1}{1!}\alpha(\tau_1) = -\frac{1}{2}$

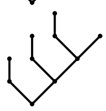
For $\mathbb{T}_2$ we find,

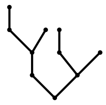
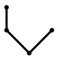
$\tau =$  $\tau_1 =$  $\alpha(\tau) = \frac{B_1}{1!}\alpha(\tau_1) = \frac{1}{4}$

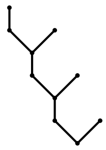
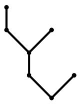
$\tau =$  $\tau_1 = \bullet$ $\tau_2 = \bullet$ $\alpha(\tau) = \frac{B_2}{2!}\alpha(\tau_1)\alpha(\tau_2) = \frac{1}{12}$

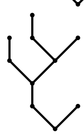
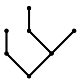
For $\mathbb{T}_3$ we find,

$\tau =$  $\tau_1 = \bullet$ $\tau_2 =$  $\alpha(\tau) = \frac{B_2}{2!}\alpha(\tau_1)\alpha(\tau_2) = -\frac{1}{24}$

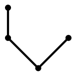
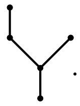
$\tau =$  $\tau_1 = \bullet$ $\tau_2 = \bullet$ $\tau_3 = \bullet$ $\alpha(\tau) = \frac{B_3}{3!}\alpha(\tau_1)\alpha(\tau_2)\alpha(\tau_3)$
$= 0$ since $B_3 = 0$

$\tau =$  $\tau_1 =$  $\tau_2 = \bullet$ $\alpha(\tau) = \frac{B_2}{2!}\alpha(\tau_1)\alpha(\tau_2) = -\frac{1}{24}$

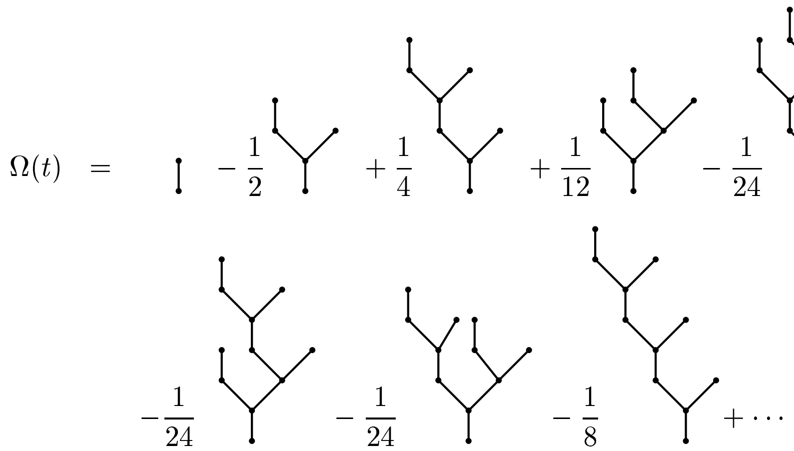
$\tau =$  $\tau_1 =$  $\alpha(\tau) = \frac{B_1}{1!}\alpha(\tau_1) = -\frac{1}{8}$

$\tau =$  $\tau_1 =$  $\alpha(\tau) = \frac{B_1}{1!}\alpha(\tau_1) = -\frac{1}{24}$

In Table 2.1, all terms in the Magnus series expansion associated with the rooted trees of up to $\mathbb{T}_3$ are listed, as well as the values of their scalar constants $\alpha(\tau)$. Note that in Eq (2.10), $C_\tau(\xi)$ is integrated once more from 0 to t. Hence, to accurately represent the terms in the Magnus series expansion, we need to add a vertical branch to the bottom root of each tree. For example, with $\tau \in \mathbb{T}_1$,

 has now become  .

Now, we can rewrite the Magnus series expansion in terms of the rooted trees, so that Eq (2.10) is given by,



(2.16)

2.3 Power m of a Rooted Tree

Now that we have the rooted tree theory for the construction of terms in the Magnus series expansion up to any k numbers of commutators, we need to find out how to construct an approximate $\Omega_p(t)$ of a specific order p, such that,

$$\Omega(t) \approx \Omega_p(t) + O(t^{p+1}).$$

In the following section, we follow the summary in [19] for the construction of the Magnus series expansion of up to order $p = 4$.

From Eq (2.11) in Section 2.2, we introduced the notation,

$$C_\tau(\xi) = \left[\int_0^\xi C_{\tau_1}(\xi_1)d\xi_1, C_{\tau_2}(\xi) \right].$$

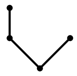
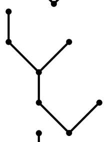
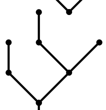
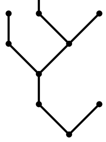
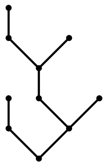
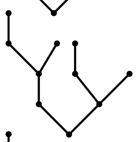
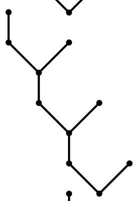
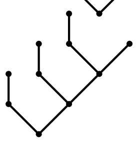
Let us suppose that a tree $\tau \in \mathbb{T}_k$ is of power m, where $m \geq 0$ is the smallest integer such that for a smooth matrix function $A(t)$ [19],

$$C_\tau(t) = O(t^m).$$

If we denote $\mathbb{F}_m$ as the set of all rooted trees of power m, then the truncated Magnus series expansion of order p is given by,

$$\Omega_p(t) = \sum_{m=0}^{p-1} \sum_{\tau \in \mathbb{F}_m} \alpha(\tau) \int_0^t C_\tau(\xi)d\xi. \tag{2.17}$$

Table 2.1: Rooted trees for the Magnus series expansion.

p	$\mathbb{T}_k$	$\mathbb{F}_m$	$C_\tau(\xi)$	τ	$\alpha(\tau)$
1	$\mathbb{T}_0$	$\mathbb{F}_0$	$A(\xi)$		1
3	$\mathbb{T}_1$	$\mathbb{F}_2$	$\int_0^{\xi_1} [A(\xi_2), A(\xi_1)]\, d\xi_2$		$-\frac{1}{2}$
4	$\mathbb{T}_2$	$\mathbb{F}_3$	$\int_0^{\xi_1}\int_0^{\xi_2} [[A(\xi_3), A(\xi_2)], A(\xi_1)]\, d\xi_3 d\xi_2$		$\frac{1}{4}$
4	$\mathbb{T}_2$	$\mathbb{F}_3$	$\int_0^{\xi_1}\int_0^{\xi_1} [A(\xi_2), [A(\xi_3), A(\xi_1)]]\, d\xi_3 d\xi_2$		$\frac{1}{12}$
5	$\mathbb{T}_3$	$\mathbb{F}_4$	$\int_0^{\xi_1}\int_0^{\xi_2}\int_0^{\xi_3} [[A(\xi_3), [A(\xi_3), A(\xi_2)]], A(\xi_1)]\, d\xi_3 d\xi_3 d\xi_2$		$-\frac{1}{24}$
5	$\mathbb{T}_3$	$\mathbb{F}_4$	$\int_0^{\xi_1}\int_0^{\xi_1}\int_0^{\xi_2} [A(\xi_2), [[A(\xi_3), A(\xi_2)], A(\xi_1)]]\, d\xi_3 d\xi_2 d\xi_2$		$-\frac{1}{24}$
7	$\mathbb{T}_3$	$\mathbb{F}_6$	$\int_0^{\xi_1}\int_0^{\xi_2}\int_0^{\xi_1} [[A(\xi_3), A(\xi_2)], [A(\xi_2), A(\xi_1)]]\, d\xi_3 d\xi_2 d\xi_2$		$-\frac{1}{24}$
5	$\mathbb{T}_3$	$\mathbb{F}_4$	$\int_0^{\xi_1}\int_0^{\xi_2}\int_0^{\xi_3} [A(\xi_2), [A(\xi_2), [A(\xi_2), A(\xi_1)]]]\, d\xi_4 d\xi_3 d\xi_2$		$-\frac{1}{8}$
5	$\mathbb{T}_3$	$\mathbb{F}_4$	$\int_0^{\xi_1}\int_0^{\xi_1}\int_0^{\xi_1} [A(\xi_2), [A(\xi_2), [A(\xi_2), A(\xi_1)]]]\, d\xi_2 d\xi_2 d\xi_2$		0

Note that in the above equation, the largest value of the power m is not equal to order p. This is because power m represents the order of the rooted tree associated with $C_\tau(\xi)$, and not the integral of $C_\tau(\xi)$. Another integration from 0 to t, must be done in Eq (2.17), so that the actual order p of $\Omega_p(t)$ is given by,

$$p = m + 1.$$

From Eq (2.15), we know that for a rooted tree $\tau \in \mathbb{T}_k$, then the same tree τ is in the set $\mathbb{F}_m$ for the power $m \geq k$. To find m for the tree τ, we need to a find a relationship between the power m for $C_\tau(t)$ and the powers m_1 and m_2 with respect to $C_{\tau_1}(t)$ and $C_{\tau_2}(t)$ in Eq (2.11). This can be done by writing $C_{\tau_i}(t)$ as a series in power $O(t^{m_i})$, such that,

$$C_{\tau_i}(t) = a_i t^{m_i} + b_i t^{m_i+1} + c_i t^{m_i+2} + \cdots \qquad \text{for i=1,2,}\ldots,$$

and then we substitute this series into Eq (2.11). If we assume that $C_{\tau_1}(t)$ is of power m_1, such that $C_{\tau_1}(t) = O(t^{m_1})$, then the integral inside the commutator in Eq (2.11) becomes,

$$\begin{aligned} \int_0^t C_{\tau_1}(\xi)d\xi &= \int_0^t (a_1\xi^{m_1} + b_1\xi^{m_1+1} + c_1\xi^{m_1+2} + \cdots)d\xi \\ &= \frac{a_1}{m_1+1}t^{m_1+1} + \frac{b_1}{m_1+2}t^{m_1+2} + \frac{c_1}{m_1+3}t^{m_1+3} + \cdots \end{aligned} \tag{2.18}$$

In this way, we can find the expansion for Eq (2.11), given that C_{τ_1} is of power m_1 and C_{τ_2} is of power m_2,

$$\begin{aligned} C_\tau(t) &= \left[\int_0^t C_{\tau_1}(\xi)d\xi, C_{\tau_2}(t)\right] \\ &= \Bigg[\frac{a_1}{m_1+1}t^{m_1+1} + \frac{b_1}{m_1+2}t^{m_1+2} + \frac{c_1}{m_1+3}t^{m_1+3} + \cdots, \\ &\qquad\qquad a_2 t^{m_2} + b_2 t^{m_2+1} + c_2 t^{m_2+2} + \cdots\Bigg] \\ &= \left(\frac{[a_1,a_2]}{m_1+1}\right)t^{m_1+m_2+1} + \left(\frac{[a_1,b_2]}{m_1+1} + \frac{[b_1,a_2]}{m_1+2}\right)t^{m_1+m_2+2} \\ &\quad + \left(\frac{[a_1,c_2]}{m_1+1} + \frac{[b_1,b_2]}{m_1+2} + \frac{[c_1,a_2]}{m_1+3}\right)t^{m_1+m_2+3} + \cdots \; = \; O\left(t^{m_1+m_2+1}\right). \end{aligned}$$

That is, for $\tau_1 \neq \tau_2$, power m of $C_\tau(t)$ is given by,

$$m = m_1 + m_2 + 1. \tag{2.19}$$

Furthermore, when $\tau_1 = \tau_2$, the power m of $C_\tau(t)$ can be improved in the following way, by performing a similar substitution into Eq (2.11),

$$\begin{aligned} C_\tau(t) &= \left[\int_0^t C_{\tau_1}(\xi)d\xi, C_{\tau_1}(t)\right] \\ &= \Bigg[\frac{a_1}{m_1+1}t^{m_1+1} + \frac{b_1}{m_1+2}t^{m_1+2} + \frac{c_1}{m_1+3}t^{m_1+3} + \cdots, \\ &\qquad\qquad a_1 t^{m_1} + b_1 t^{m_1+1} + c_1 t^{m_1+2} + \cdots\Bigg] \end{aligned}$$

$$
\begin{aligned}
&= \frac{a_1^2 - a_1^2}{m_1+1} t^{2m_1+1} + \frac{a_1 b_1 - b_1 a_1}{(m_1+1)(m_1+2)} t^{2m_1+2} + \frac{2(a_1 c_1 - c_1 a_1)}{(m_1+1)(m_1+3)} t^{2m_1+3} + \cdots \\
&= \frac{[a_1, b_1]}{(m_1+1)(m_1+2)} t^{2m_1+2} + \frac{2[a_1, c_1]}{(m_1+1)(m_1+3)} t^{2m_1+3} + \cdots \; = \; O\left(t^{2m_1+2}\right).
\end{aligned}
$$

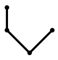
That is, for $\tau_1 = \tau_2$, the power m of $C_\tau(t)$ is given by,

$$m = 2m_1 + 2. \tag{2.20}$$

To group the rooted trees into the sets $\mathbb{F}_m$ of power m, we first make the identification that the tree of single vertex representing $A(\xi)$ has power $m = 0$,

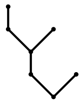
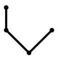
$$\tau = \; \cdot \; \in \mathbb{F}_0, \qquad \text{where } m = 0.$$

Starting with $k = 1$, we have the tree τ with sub-trees τ_1 and τ_2 according to Eq (2.12),

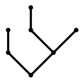
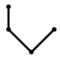
$\tau =$  $\in \mathbb{T}_1, \qquad$ with $\tau_1 = \tau_2 = \cdot$

Note that in this case, $\tau_1 = \tau_2$ and the powers $m_1 = m_2 = 0$. So by Eq (2.20), the power m of $C_\tau(t)$ when $\tau \in \mathbb{T}_1$ is $m = 2m_1 + 2 = 2$.

When $k = 2$, we have two trees in $\mathbb{T}_2$. The first one is given by,

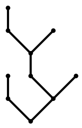
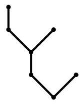
$\tau =$  $\in \mathbb{T}_2 \qquad$ with $\tau_1 =$  $\in \mathbb{F}_2$ and $\tau_2 = \cdot \in \mathbb{F}_0$.

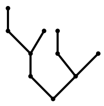
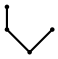
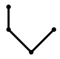
So that the power m of τ is $m = m_1 + m_2 + 1 = 2 + 0 + 1 = 3$. Similarly, the second tree in $\mathbb{T}_2$ is,

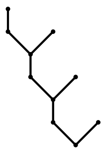
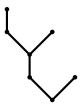
$\tau =$  $\in \mathbb{T}_2 \qquad$ with $\tau_1 = \cdot \in \mathbb{F}_0$ and $\tau_2 =$  $\in \mathbb{F}_2$.

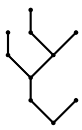
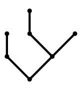
So the power m of τ is $m = m_1 + m_2 + 1 = 0 + 2 + 1 = 3$.

When $k = 3$, we find that,

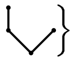
$\tau =$  $\in \mathbb{T}_3 \quad \tau_1 = \cdot \in \mathbb{F}_0 \quad \tau_2 =$  $\in \mathbb{F}_3 \quad m = 4$

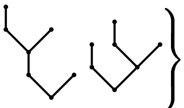
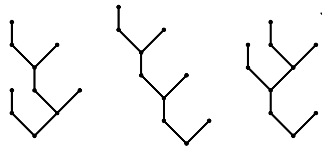
$\tau =$  $\in \mathbb{T}_3 \quad \tau_1 =$  $\in \mathbb{F}_2 \quad \tau_2 =$  $\in \mathbb{F}_2 \quad m = 6$

$\tau =$  $\in \mathbb{T}_3 \quad \tau_1 =$  $\in \mathbb{F}_3 \quad \tau_2 = \cdot \in \mathbb{F}_0 \quad m = 4$

$\tau =$  $\in \mathbb{T}_3 \quad \tau_1 =$  $\in \mathbb{F}_3 \quad \tau_2 = \cdot \in \mathbb{F}_0 \quad m = 4$

Hence, for power m up to $m = 4$, the set $\mathbb{F}_m$ contains the following trees, which are of order $p = m + 1$ in the Magnus series expansion,

$\mathbb{F}_0 = \{\ \cdot\ \} (p = 1)$ $\qquad$ $\mathbb{F}_1 = \{\ \}$ $\qquad$ $\mathbb{F}_2 = \{$  $\}$ $(p = 3)$

$\mathbb{F}_3 = \{$  $\}$ $(p = 4)$ $\qquad$ $\mathbb{F}_4 = \{$  $\}$ $(p = 5)$

Now that we have the method of building trees for all the terms in the approximation to $\Omega(t)$ up to any order p, it was discovered that the linear combination of the same order trees listed above can be of higher order still. Hence this leads to a further reduction of the number of terms needed for the Magnus series expansion of a particular order p. We will look at this in further detail in the next section.

2.4 Time-Symmetry and Order of the Magnus Series Expansion

From Section 2.3, we know that rooted trees belonging to the set $\mathbb{F}_{m+1}$ of power $m + 1$ are also of order $p = m + 1$. Hence, it would seem that to construct a Magnus series expansion of order p, we need to include all of the terms associated with trees from all sets of power up to $\mathbb{F}_{m+1}$. However, the number of terms can be further reduced if we consider the time-symmetry of the method [19].

Let $y(t) = \Psi_t(y_0)$ be the flow of the differential equation in Eq (2.1). The flow is *time-symmetric* (i.e. $\Psi_{-t} \circ \Psi_t = I$), because integrating from 0 to t and back to 0 yields the original initial value.

Furthermore, it has been proved in [21] that the truncation by order p of the Magnus series expansion given in Eq (2.17) also respects time symmetry. In other words, if the truncated flow has the form, $\widetilde{\Psi}_t(y_0) = \exp(\Omega_p(t))y_0$, for $t \geq 0$, then $\widetilde{\Psi}_{-t} \circ \widetilde{\Psi}_t = I$. Also, any analytic time-symmetric map S_t can be represented in the form $S_t = \exp(C(t))$, where $C(t)$ is an expansion in *odd powers of t only* [19]. Combining all of the above, we have the key result which reduces the number of terms in the Magnus series expansion.

Theorem 2.4. *The function $\Omega_p(t)$ can be expanded in odd powers of t and*

$$\Omega(t) = \Omega_{2q-1}(t) + O(t^{2q+1}), \qquad q \in \mathbb{Z}_+.$$

Theorem 2.4 implies that for a Magnus expansion of order 4, only the terms in the sets up to $\mathbb{F}_2$ are needed. This is because the linear combination of trees from $\mathbb{F}_3$ is actually $O(t^5)$, and hence does not affect an order 4 approximation of $\Omega(t)$. (i.e. For $q = 2$ in Theorem 2.4 we find $\Omega(t) = \Omega_3(t) + O(t^5)$. So $\Omega_3(t)$ is in fact an order 4 approximation.) According to Theorem 2.4, we can state the terms in the Magnus series expansion up to order 6 in the following way,

$$
\begin{aligned}
\Omega(t) \quad = \quad & \int_0^t A(\xi_1)d\xi_1 \qquad \rightarrow \text{order } 2 & (2.21)\\
& -\frac{1}{2}\int_0^t\int_0^{\xi_1} \left[A(\xi_2), A(\xi_1)\right] d\xi_2 d\xi_1 \qquad \rightarrow \text{order4} & (2.22)\\
& +\frac{1}{4}\int_0^t\int_0^{\xi_1}\int_0^{\xi_2} \left[\left[A(\xi_3), A(\xi_2)\right], A(\xi_1)\right] d\xi_3 d\xi_2 d\xi_1 \\
& +\frac{1}{12}\int_0^t\int_0^{\xi_1}\int_0^{\xi_1} \left[A(\xi_2), \left[A(\xi_3), A(\xi_1)\right]\right] d\xi_3 d\xi_2 d\xi_1 \\
& -\frac{1}{24}\int_0^t\int_0^{\xi_1}\int_0^{\xi_2}\int_0^{\xi_2} \left[\left[A(\xi_3), \left[A(\xi_3), A(\xi_2)\right]\right], A(\xi_1)\right] d\xi_3 d\xi_3 d\xi_2 d\xi_1 \\
& -\frac{1}{24}\int_0^t\int_0^{\xi_1}\int_0^{\xi_1}\int_0^{\xi_2} \left[A(\xi_2), \left[\left[A(\xi_3), A(\xi_2)\right], A(\xi_1)\right]\right] d\xi_3 d\xi_2 d\xi_2 d\xi_1 \\
& -\frac{1}{8}\int_0^t\int_0^{\xi_1}\int_0^{\xi_2}\int_0^{\xi_3} \left[\left[\left[A(\xi_4), A(\xi_3)\right], A(\xi_2)\right]\right] d\xi_4 d\xi_3 d\xi_2 d\xi_1 \rightarrow \text{order } 6 & (2.23)\\
& +\cdots
\end{aligned}
$$

For an explicit example with a matrix differential equation, please refer to the *Airy equation* in [18].

Given that we can determine all the terms in the Magnus series expansion up to order p, we now need to discuss ways of computing such terms numerically for the matrix function $A(t)$. The technique of solving multiple integrals is known as multivariate quadrature, which is the topic in the next section.

2.5 Multivariate Quadrature

From the previous sections, we see the need for using some sort of quadrature technique to evaluate the multiple integrals arising in the terms for the Magnus series expansion $\Omega_p(t)$. Before we discuss the details of how to approximate such integrals using multivariate quadrature, let us review some of the important details concerning Gaussian quadrature of an integral in one dimension.

2.5.1 Gauss-Legendre Quadrature

The Legendre polynomial of degree n, denoted $P_n(x)$, is usually defined on the symmetric interval $[-1, 1]$. However, if we let shift the interval to $[0, 1]$, then the Legendre polynomial, now denoted $P_n^*(x)$ is shifted by $2x - 1$, so that $P_n^*(x) = P_n(2x - 1)$. The first few $P_n^*(x)$ are defined to be [7],

$$\begin{aligned} P_0^*(x) &= 1 \\ P_1^*(x) &= 2x - 1 \\ P_2^*(x) &= 6x^2 - 6x + 1 \\ P_3^*(x) &= 20x^3 - 30x^2 + 12x - 1 \\ &\vdots \end{aligned}$$

The following lemma presents some of the properties of the Legendre polynomials which we will use in the discussion about Gaussian quadrature later on.

Lemma 2.7. *There exist polynomials $P_n^* : [0,1] \to \mathbb{R}$, of degrees n, for $n = 0, 1, 2, \dots$ with the properties that,*

$$\int_0^1 P_m^*(x) P_n^*(x) dx = 0, \qquad m \neq n$$

$P_n^*(x)$ has n distinct real zeros in the open interval $(0,1)$ for $n = 0, 1, 2, \dots$.

For the outline of the proof of Lemma 2.7, please refer to [7] or [5].

The idea behind the quadrature formula is to approximate the integral of a function of a single variable. This is done by finding a weighted sum of the function evaluated at certain points (or nodes) in the integration interval. Approximations like the Newton-Cotes formulae use values of the function at equally spaced points [5]. But in the case of the Gaussian quadrature, the points for the function evaluation are chosen for optimal accuracy. That is, the nodes c_i for $i = 1, 2, 3, \dots$ are chosen to minimise the expected error obtained in performing the approximation with weighted sums. It turns out the points of function evaluations are conveniently given by the zeros of the Legendre polynomials $P_n^*(x)$ for interval $[0,1]$. Furthermore, the integral approximation gives exact results for any polynomial of degree less than $2s$ if the roots of a s-degree Legendre polynomial $P_s^*(x)$ are used. The following theorem states the Gaussian quadrature formula for the approximation of a single integral [5].

Theorem 2.5. *Suppose that c_i, for $i = 1, 2, \dots, s$ are the roots of an s-degree Legendre polynomial $P_s^*(x)$, and that for each $i = 1, 2, \dots, s$ the weights b_i are defined by,*

$$b_i = \int_0^1 \prod_{\substack{j=1 \\ j \neq i}}^{s} \frac{x - c_j}{c_i - c_j} dx.$$

If $\phi(x)$ is any polynomial of degree less than $2s$, then the Gauss-Legendre quadrature is given by,

$$\int_0^1 \phi(x) dx = \sum_{i=1}^{s} b_i \phi(c_i).$$

Proof. Recall that an nth degree Lagrange interpolating polynomial is a unique polynomial K of degree at most n for $n+1$ distinct interpolation points c_i, for $i = 1, 2, \dots, n+1$. The unique polynomial K which interpolates the function f is given by [5],

$$K(x) = \sum_{i=1}^{n+1} f(c_i) \cdot \prod_{\substack{j=1 \\ j \neq i}}^{n+1} \frac{x - c_j}{c_i - c_j},$$

with the property that,

$$f(c_i) = K(c_i), \quad \text{for i=1,2,...,n+1.}$$

Now, for a polynomial $R(x)$ of degree $< s$, let us write $R(x)$ as an $s-1$ degree Lagrange polynomial $K_{s-1}(x)$, with nodes $c_1, c_2, \ldots, c_s$ at the s roots of an $s-1$ degree Legendre polynomial $P^*_{s-1}(x)$ in the interval $[0,1]$. Integrating this Lagrange polynomial yields,

$$\begin{aligned}\int_0^1 R(x)dx &= \int_0^1 \left[\sum_{i=1}^{s} R(c_i) \prod_{\substack{j=1\\ j\neq i}}^{s} \frac{x-c_j}{c_i-c_j}\right] dx = \sum_{i=1}^{s}\left[\int_0^1 \prod_{\substack{j=1\\ j\neq i}}^{s} \frac{x-c_j}{c_i-c_j} dx\right] R(c_i) \\ &= \sum_{i=1}^{s} b_i R(c_i).\end{aligned}$$

This shows that the theorem is true for the polynomial $R(x)$ of degree $< s$.

Let us suppose that we now consider a polynomial $\phi(x)$ of degree $< 2s$. We divide $\phi(x)$ by a Legendre polynomial $P^*_s(x)$ of degree s. This yields two polynomials $Q(x)$ and $R(x)$ of degree $< s$, such that, $\phi(x) = Q(x)P^*_s(x) + R(x)$. Also, recall that the Legendre polynomials $P^*_s(x)$ are orthogonal polynomials by Lemma 2.7. Since $Q(x)$ is of degree $< s$, then,

$$\int_0^1 Q(x)P^*_s(x)dx = 0.$$

Next, since c_i for $i = 1, \ldots, s$ are the zeros of $P^*_s(x)$, then,

$$\phi(c_i) = Q(c_i)P^*_s(c_i) + R(c_i) = R(c_i).$$

Finally, we integrate the polynomial $\phi(x)$ from 0 to 1, and putting all of the above together,

$$\begin{aligned}\int_0^1 \phi(x)dx &= \int_0^1 \left[Q(x)P^*_s(x) + R(x)\right] dx = 0 + \int_0^1 R(x)dx \\ &= \sum_{i=1}^{s} b_i R(c_i) = \sum_{i=1}^{s} b_i \phi(c_i).\end{aligned}$$

□

As an interesting note, the concept presented in Theorem 2.5 can be taken one step further, in the sense that we can consider the possibility of constructing an s stage implicit Runge-Kutta method with order $2s$ [7]. These methods do exist and are known as the Gauss methods. A recipe for constructing such methods is presented in [7], where the values c_i and b_i are already given by Theorem 2.5. The coefficient a_{ij} in the Butcher Tableau can then be found by satisfying the order conditions for the Runge-Kutta method of stage s. For further details please refer to [7]. One example of such a Gauss method (with $s = 2$ and $p = 4$) is presented below, and this tableau is used to construct one of the numerical integrators for solving some of the differential equations in Chapter 4.

$$\begin{array}{c|cc} \frac{1}{2}-\frac{\sqrt{3}}{6} & \frac{1}{4} & \frac{1}{4}-\frac{\sqrt{3}}{6} \\ \frac{1}{2}+\frac{\sqrt{3}}{6} & \frac{1}{4}+\frac{\sqrt{3}}{6} & \frac{1}{4} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

2.5.2 Multivariate Gauss-Legendre Quadrature

Following the discussion on the Gauss-Legendre quadrature in Section 2.5.1, we can now look at the detail of applying multivariate quadrature to find the numerical values for terms in the truncated Magnus series expansion of order p. Similarly to the one dimensional integration, the quadrature for the multiple dimensional integration is given by the weighted sum of multiple variable function evaluations. As a result, the weight b_i is now also given by the multiple integration of the product of Lagrange interpolation polynomials. We will follow the discussion presented in [19] for more details of this multivariate quadrature process.

To begin with, let us note that each integral in the Magnus series expansion is of the form,

$$I(h) = \int_S L(A(\xi_1), A(\xi_2), \ldots, A(\xi_s)) d\xi_s \ldots d\xi_1, \tag{2.24}$$

where L is a multiple variable function, s is the number of integrals in the expression, and h is the stepsize used in the discretisation of the multiple integral. Also, the integration region S is a polytope of the form,

$$S = \{\xi_1, \ldots, \xi_s \in \mathbb{R} : \xi_1 \in [0, h], \xi_l \in [0, \xi_{m_l}], l = 2, 3, \ldots, s\},$$

where $m_l \in \{1, 2, \ldots, l-1\}$, for $l = 2, 3, \ldots, s$.

To discretise Eq (2.24), we choose ν distinct quadrature points $c_1, c_2, \ldots, c_\nu \in [0, 1]$. That is, they are the roots of a Legendre polynomial $P^*_\nu(x)$ of degree ν. We then calculate the approximation $A_k = hA(c_k h)$, for $k = 1, 2, \ldots, \nu$ and form the quadrature,

$$D(h) = \sum_{\mathbf{k} \in C^\nu_s} b_{\mathbf{k}} L(A_{k_1}, A_{k_2}, \ldots, A_{k_s}), \tag{2.25}$$

where $k_1, k_2, \ldots, k_s \in \mathbf{k}$ and C^ν_s is the set of all combinations of s-tuples $\mathbf{k}$ from the set $\{1, 2, \ldots, \nu\}$, excluding s repeats of the same number in the set.

Similarly to the weights b_i in Theorem 2.5, the weights in this case are given by,

$$b_{\mathbf{k}} = \int_{\widetilde{S}} \prod_{i=1}^{s} l_{k_i}(\xi_i) d\xi_i, \tag{2.26}$$

where,

$$\widetilde{S} = \{\xi_1, \ldots, \xi_s \in \mathbb{R} : \xi_1 \in [0, 1], \xi_l \in [0, \xi_{m_l}], l = 2, 3, \ldots, s\}$$

is the original polytope S scaled to a unit cube. Note that the function $l_j(x)$ is the cardinal Lagrange interpolation polynomial in Section 2.5.1,

$$l_j(x) = \prod_{\substack{i=1 \\ i \neq j}}^{\nu} \frac{x - c_i}{c_j - c_i}, \qquad j = 1, 2, \ldots, \nu. \tag{2.27}$$

Note that because k_i are elements in the s-tuple $\mathbf{k}$, Eq (2.26) is actually a multiple integral over the product of s cardinal Lagrange polynomials $l_{k_i}(\xi_i)$. Furthermore, we can derive the expression for $b_{\mathbf{k}}$ in a similar way as in the proof for Theorem 2.5. If we let the following be the Lagrange interpolation polynomial at the roots c_i of the Legendre polynomials (with scaling h),

$$K(t) = \sum_{k=1}^{\nu} l_k\left(\frac{t}{h}\right)A(c_k h) = \sum_{k=1}^{\nu}\left(\sum_{\substack{i=1\\k\neq i}}^{\nu}\frac{t/h - c_i}{c_k - c_i}\right)A(c_k h). \tag{2.28}$$

Then we can substitute Eq (2.28) as the function $A(t)$ in Eq (2.24) and explicitly integrate.

Now that we know how to formulate the multivariate quadrature formula in Eq (2.25) for approximating the multiple integral given by Eq (2.24), we need to also know the order of this approximation. This is given by the following theorem [19].

Theorem 2.6. *Let $m \geq 0$ be the largest integer so that the polynomial*

$$c(t) = \prod_{k=1}^{\nu}(t - c_k)$$

obeys the following orthogonal condition,

$$\int_0^1 \xi^{i-1}c(\xi)d\xi = 0, \qquad i = 1, 2, \ldots, m.$$

The orthogonality condition implies that the quadrature formula in Eq (2.25) is of order $\nu + m$ for all polytope S and all multiple variable functions L. In particular, if $c_1, c_2, \ldots, c_\nu$ are the roots of the Legendre polynomial P_ν^* defined in Section 2.5.1 for the interval $[0, 1]$, then the quadrature in Eq (2.25) is of order 2ν.

To conclude this section on quadrature, we show how to use Eq (2.25) and Eq (2.26) to numerically construct the multivariate quadrature approximation for the first few terms in the Magnus series expansion. In particular, we want to build an algorithm for the truncated Magnus series expansion of order $p = 4$. Recall that from Theorem 2.6, if we use the Gauss-Legendre quadrature, then we only need to use the roots of the Legendre polynomial P_ν^* of degree $\nu = 2$ for a 4th order approximation.

Back in Section 2.4 we displayed the terms in the Magnus series expansion up to order 6. Let us relabel the multiple integrals as $I_i(t)$ for $i = 1, 2, \ldots$, so we now have,

$$\Omega(t) = I_1 - \frac{1}{2}I_2 + \frac{1}{12}I_3 + \frac{1}{4}I_4 + \cdots,$$

where the first four terms are listed below.

$$\begin{aligned}
I_1(t) &= \int_0^t A(\xi_1)d\xi_1 \\
I_2(t) &= \int_0^t\int_0^{\xi_1} [A(\xi_2), A(\xi_1)]\, d\xi_2 d\xi_1 \\
I_3(t) &= \int_0^t\int_0^{\xi_1}\int_0^{\xi_1} [A(\xi_2), [A(\xi_3), A(\xi_1)]]\, d\xi_3 d\xi_2 d\xi_1 \\
I_4(t) &= \int_0^t\int_0^{\xi_1}\int_0^{\xi_2} [[A(\xi_3), A(\xi_2)], A(\xi_1)]\, d\xi_3 d\xi_2 d\xi_1
\end{aligned}$$

We want to use the multivariate quadrature formulae to approximate the first four integrals $I_i(t)$, $i = 1, \ldots, 4$ in the Magnus series expansion. The details are given below.

First, the roots in the Legendre polynomial $P_2^*(x)$ of degree $\nu = 2$ are $c_1 = \frac{1}{2} - \frac{\sqrt{3}}{6}$ and $c_2 = \frac{1}{2} + \frac{\sqrt{3}}{6}$. So, the cardinal Lagrange interpolation polynomials in Eq (2.27) are,

$$l_1(x) = \frac{x - c_2}{c_1 - c_2} = -\sqrt{3}x + \frac{1}{2}(\sqrt{3}+1) \tag{2.29}$$

$$l_2(x) = \frac{x - c_1}{c_2 - c_1} = \sqrt{3}x - \frac{1}{2}(\sqrt{3}-1). \tag{2.30}$$

Furthermore, the approximations for the function $A(t)$ at the two roots c_1, c_2 are given by $A_1 = hA((\frac{1}{2} - \frac{\sqrt{3}}{6})h)$ and $A_2 = hA((\frac{1}{2} + \frac{\sqrt{3}}{6})h)$.

For the multivariate quadrature of the first integral $I_1(t)$, the function L is $L(A(\xi_1)) = A(\xi_1)$, and the set of combinations of s tuples values is given by $C_s^\nu = C_1^2 = \{(1), (2)\}$. The weights $b_{\mathbf{k}}$ can now be found by integration according to Eq (2.26), as,

$$b_{(1)} = \int_0^1 l_1(\xi_1) d\xi_1 = \frac{1}{2} \qquad b_{(2)} = \int_0^1 l_2(\xi_1) d\xi_1 = \frac{1}{2},$$

which is the same as the calculation to find the weights b_i in Theorem 2.5 for the Gauss-Legendre quadrature of single integrals. Hence, the multivariate quadrature using Gauss-Legendre points for $I_1(t)$ is given by Eq (2.25),

$$D_1(h) = b_{(1)} L(A_1) + b_{(2)} L(A_2) = \frac{1}{2}A_1 + \frac{1}{2}A_2. \tag{2.31}$$

For the multivariate quadrature of the second integral $I_2(t)$, the function L and the combination of $s = 2$ tuples of numbers in set C_s^ν are given by,

$$L(A(\xi_1), A(\xi_2)) = [A(\xi_2), A(\xi_1)], \qquad C_s^\nu = C_2^2 = \{(1,2), (2,1)\}.$$

So, the weights $b_{\mathbf{k}}$ are,

$$b_{(1,2)} = \int_0^1 \int_0^{\xi_1} l_1(\xi_1) d\xi_1 l_2(\xi_2) d\xi_2 = \frac{1}{8} - \frac{\sqrt{3}}{12} \qquad b_{(2,1)} = \int_0^1 \int_0^{\xi_1} l_2(\xi_1) d\xi_1 l_1(\xi_2) d\xi_2 = \frac{1}{8} + \frac{\sqrt{3}}{12},$$

and the quadrature for $I_2(t)$ is given by,

$$\begin{aligned} D_2(h) &= b_{(1,2)} L(A_1, A_2) + b_{(2,1)} L(A_2, A_1) = b_{(1,2)}[A_2, A_1] + b_{(2,1)}[A_1, A_2] \\ &= (b_{(1,2)} - b_{(2,1)})[A_2, A_1] = -\frac{\sqrt{3}}{6}[A_2, A_1]. \end{aligned} \tag{2.32}$$

The multivariate quadrature of the third integral $I_3(t)$ is a little more complicated. As in the above two examples, the function L and the combination of $s = 3$ tuples of numbers in set C_s^ν are given by,

$$\begin{aligned} L(A(\xi_1), A(\xi_2), A(\xi_3)) &= [A(\xi_2), [A(\xi_3), A(\xi_1)]] \\ C_s^\nu = C_3^2 &= \left\{ \begin{array}{ccc} (1,1,2) & (1,2,1) & (2,1,1) \\ (2,2,1) & (2,1,2) & (1,2,2) \end{array} \right\}. \end{aligned}$$

For each 3-tuple of numbers in C_3^2, we can find the corresponding weights $b_{\mathbf{k}}$, such that for the first tuple $(1,1,2)$, we integrate Eq (2.26) to find,

$$b_{(1,1,2)} = \int_0^1 \int_0^{\xi_1} \int_0^{\xi_1} l_1(\xi_1)d\xi_1 l_1(\xi_2)d\xi_2 l_2(\xi_3)d\xi_3 = -\frac{\sqrt{3}}{48} + \frac{7}{240}.$$

Integrating for each $b_{\mathbf{k}}$ separately yields,

$$\begin{array}{lll} b_{(1,1,2)} = -\frac{\sqrt{3}}{48} + \frac{7}{240} & b_{(1,2,1)} = -\frac{\sqrt{3}}{48} + \frac{7}{240} & b_{(2,1,1)} = \frac{\sqrt{3}}{24} + \frac{1}{15} \\ b_{(2,2,1)} = \frac{\sqrt{3}}{48} + \frac{7}{240} & b_{(2,1,2)} = \frac{\sqrt{3}}{48} + \frac{7}{240} & b_{(1,2,2)} = -\frac{\sqrt{3}}{24} + \frac{1}{15}. \end{array}$$

Also, note that because the function L involves two nested commutators for $I_3(t)$. Before we form the sum for the corresponding quadrature $D_3(h)$, the number of function evaluations of L can be reduced from 6 separate L evaluations down to 2 sets of nested commutators, due to skew-symmetry in the commutator.

$$\begin{aligned} L(A_1, A_1, A_2) &= [A_1, [A_2, A_1]] = -[[A_2, A_1], A_1] \\ L(A_1, A_2, A_1) &= [A_2, [A_1, A_1]] = 0 \\ L(A_2, A_1, A_1) &= [A_1, [A_1, A_2]] = [A_1, -[A_2, A_1]] = [[A_2, A_1], A_1] \\ L(A_2, A_2, A_1) &= [A_2, [A_1, A_2]] = [A_2, -[A_2, A_1]] = [[A_2, A_1], A_2] \\ L(A_2, A_1, A_2) &= [A_1, [A_2, A_2]] = 0 \\ L(A_1, A_2, A_2) &= [A_2, [A_2, A_1]] = -[[A_2, A_1], A_2] \end{aligned}$$

Now, putting all of the above together to form the quadrature $D_3(h)$, we find that,

$$\begin{aligned} D_3(h) &= (b_{(2,1,1)} - b_{(1,1,2)})[[A_2, A_1], A_1] + (b_{(2,2,1)} - b_{(1,2,2)})[[A_2, A_1], A_2] \\ &= \left(\frac{\sqrt{3}}{16} + \frac{3}{80}\right)[[A_2, A_1], A_1] + \left(\frac{\sqrt{3}}{16} - \frac{3}{80}\right)[[A_2, A_1], A_2]. \end{aligned} \tag{2.33}$$

Similarly for the fourth integral $I_4(t)$, we have the function L and C_s^ν,

$$\begin{aligned} L(A(\xi_1), A(\xi_2), A(\xi_3)) &= [[A(\xi_3), A(\xi_2)], A(\xi_1)] \\ C_s^\nu = C_3^2 &= \left\{ \begin{array}{lll} (1,1,2) & (1,2,1) & (2,1,1) \\ (2,2,1) & (2,1,2) & (1,2,2) \end{array} \right\}. \end{aligned}$$

For each 3-tuple of numbers in C_3^2, we can find the corresponding weights $b_{\mathbf{k}}$, such that for the first tuple $(1,1,2)$, we integrate Eq (2.26) to find,

$$b_{(1,1,2)} = \int_0^1 \int_0^{\xi_1} \int_0^{\xi_2} l_1(\xi_1)d\xi_1 l_1(\xi_2)d\xi_2 l_2(\xi_3)d\xi_3 = -\frac{\sqrt{3}}{48} + \frac{1}{30}.$$

Integrating for each $b_{\mathbf{k}}$ separately yields,

$$\begin{array}{lll} b_{(1,1,2)} = -\frac{\sqrt{3}}{48} + \frac{1}{30} & b_{(1,2,1)} = -\frac{1}{240} & b_{(2,1,1)} = \frac{\sqrt{3}}{48} + \frac{1}{30} \\ b_{(2,2,1)} = \frac{\sqrt{3}}{48} + \frac{1}{30} & b_{(2,1,2)} = -\frac{1}{240} & b_{(1,2,2)} = -\frac{\sqrt{3}}{48} + \frac{1}{30}. \end{array}$$

and the function evaluations L corresponding to each $b_{\mathbf{k}}$ are,

$$\begin{aligned}
L(A_1,A_1,A_2) &= [[A_2,A_1],A_1] = -[A_1,[A_2,A_1]] \\
L(A_1,A_2,A_1) &= [[A_1,A_2],A_1] = [-[A_2,A_1],A_1] = [A_1,[A_2,A_1]] \\
L(A_2,A_1,A_1) &= [[A_1,A_1],A_2] = 0 \\
L(A_2,A_2,A_1) &= [[A_1,A_2],A_2] = [-[A_2,A_1],A_2] = [A_2,[A_2,A_1]] \\
L(A_2,A_1,A_2) &= [[A_2,A_1],A_2] = -[A_2,[A_2,A_1]] \\
L(A_1,A_2,A_2) &= [[A_2,A_2],A_1] = 0.
\end{aligned}$$

Putting all of the above together to form the quadrature $D_4(h)$ for $I_4(t)$,

$$\begin{aligned}
D_4(h) &= (b_{(1,2,1)} - b_{(1,1,2)})[A_1,[A_2,A_1]] + (b_{(2,2,1)} - b_{(2,1,2)})[A_2,[A_2,A_1]] \\
&= \left(\frac{\sqrt{3}}{48} - \frac{3}{80}\right)[A_1,[A_2,A_1]] + \left(\frac{\sqrt{3}}{48} + \frac{3}{80}\right)[A_2,[A_2,A_1]]. \qquad (2.34)
\end{aligned}$$

Hence, substituting the quadrature formulae in Eq (2.31) to Eq (2.34) to form the approximation to the Magnus series expansion $\Omega(t)$ yields,

$$\begin{aligned}
\Omega(t) &\approx D_1(h) - \frac{1}{2}D_2(h) + \frac{1}{12}D_3(h) + \frac{1}{4}D_4(h) + \cdots \\
&\approx \frac{1}{2}(A_1 + A_2) + \frac{\sqrt{3}}{12}[A_2,A_1] + \frac{1}{80}[[A_2,A_1],A_1] - \frac{1}{80}[[A_2,A_1],A_2] + \cdots
\end{aligned}$$

However, note that by Theorem 2.4 in Section 2.4, we know that the integrals corresponding to $I_3(t)$ and $I_4(t)$ are each of order 4, but their linear combination is of order 5. So they do not affect the accuracy if our approximation to $\Omega(t)$ is of order 4 only. Hence, we can ignore the last two terms in the above expansion and we are then left with,

$$\Omega(t) = \frac{1}{2}(A_1 + A_2) + \frac{\sqrt{3}}{12}[A_2,A_1] + O(t^5).$$

In summary, we can construct a one-step algorithm for the solution y of the differential equation in Eq (2.1), using the truncated Magnus series expansion of order 4 in the following way [20],

For one step of stepsize h from t_n to t_{n+1}, with $y(t_n) = y_n$,

$$\begin{aligned}
A_1 &= hA\left(t_n + \left(\frac{1}{2} - \frac{\sqrt{3}}{6}\right)h\right) \\
A_2 &= hA\left(t_n + \left(\frac{1}{2} + \frac{\sqrt{3}}{6}\right)h\right) \\
\Omega_4 &= \frac{1}{2}(A_1 + A_2) + \frac{\sqrt{3}}{12}[A_2,A_1] \\
y_{n+1} &= \exp(\Omega_4)y_n.
\end{aligned}$$

For the construction of order 6 methods using the truncated Magnus series expansion, please refer to [19] or [20]. In Chapter 4, we will show some of the numerical experiments performed using the Magnus series expansion of order 4 presented above.

CHAPTER 3

The RKMK Algorithm

In this chapter we study a class of Lie group methods known as the Runge-Kutta-Munthe-Kaas (RKMK) methods. This is an important class of methods because it allows us to use the classical Runge-Kutta (RK) methods to indirectly solve the differential equation on the homogeneous space, while preserving certain invariants in the original system. This means that the numerical solutions will always adhere to a specific manifold $\mathcal{M}$ - on which the differential equation evolves.

To begin with, we will follow the details in Engø's paper [11], where the ideas of equivariant maps and relatedness are discussed in coordinate-free formulation. Munthe-Kaas, in his original derivation of the RKMK method [28], used the exponential map to transform elements in the Lie algebra to its Lie group. But it is important to note that other mappings from $\mathfrak{g} \to G$ can also be used [11].

After this, we will concentrate on Munthe-Kaas's RKMK Algorithm as presented in [28], to show explicitly how the classical RK method is incorporated into this Lie algebra setting. Also, we will study the derivation of the order conditions for the RKMK method using the Lie-Butcher series [27], and in particular, how choosing the manifold $\mathcal{M}$ as $\mathbb{R}^n$ results in the classical RK order conditions.

3.1 Geometric Integrators in the RKMK class

In this section, we will follow closely the definitions and theorems found in [11]. They lead to the theory behind what is now known as the Runge-Kutta-Munthe-Kaas (RKMK) algorithm.

First, we will look at the idea of an equivariant map with respect to two Lie group actions Λ_1 and Λ_2 on manifolds $\mathcal{M}$ and $\mathcal{N}$. Further details on manifolds, Lie group G, and Lie group action Λ are given in Chapter 1.

Definition 3.14. *Let $\mathcal{M}$ and $\mathcal{N}$ be two manifolds, and let G be a Lie group acting on a point $x \in \mathcal{M}$, by the Lie group action $\Lambda_1 : G \times \mathcal{M} \to \mathcal{M}$, and also acting on a point $y \in \mathcal{N}$ by $\Lambda_2 : G \times \mathcal{N} \to \mathcal{N}$. A smooth map $f : \mathcal{M} \to \mathcal{N}$ is an equivariant map with respect to the actions Λ_1 and Λ_2 if, for all $g \in G$,*

$$f(\Lambda_1(g,x)) = \Lambda_2(g, f(x)), \qquad \textit{where } x \in \mathcal{M} \textit{ and } y = f(x) \in \mathcal{N}.$$

That is, Figure 3.1 commutes for all $g \in G$, so that the actions of the Lie group G on both $\mathcal{M}$ and $\mathcal{N}$ are related by the equivariant map f.

$$\begin{tikzcd}
\mathcal{M} \arrow[r, "f : \mathcal{M} \to \mathcal{N}"] & \mathcal{N} \\
\mathcal{M} \arrow[u, "\Lambda_1(g,\cdot)"] \arrow[r, "f : \mathcal{M} \to \mathcal{N}"'] & \mathcal{N} \arrow[u, "\Lambda_2(g,\cdot)"']
\end{tikzcd}$$

Figure 3.1: f is an equivariant map for all $g \in G$.

The main idea behind RKMK method is to choose the manifold $\mathcal{M}$ in Figure 3.1 to be a vector space so that we can use the RK methods. This means we need to find the equivariant map f between the vector space and the manifold $\mathcal{N}$. It so happens that the Lie algebra $\mathfrak{g}$ associated with a Lie group G is a vector space, as discussed in Section 1.3.2. Hence, if $\mathcal{M}$ is replaced by $\mathfrak{g}$ in Figure 3.1, then it is natural to think of the equivariant map $f : \mathfrak{g} \to \mathcal{N}$, with the Lie group action $\Lambda_2 : G \times \mathcal{N} \to \mathcal{N}$ on the manifold $\mathcal{N}$, and another action, say $B_g : \mathfrak{g} \to \mathfrak{g}$, which takes an element in $\mathfrak{g}$ and maps it back to $\mathfrak{g}$, for all elements g in the Lie group G. Such action B_g is defined in the following lemma, where Ψ is the coordinate map from $\mathfrak{g}$ to G.

Lemma 3.8. *Let $\Psi : \mathfrak{g} \to G$ be a local map, and for all $g \in G$ there exists a left multiplication $L_g : G \to G$. Then the local action B_g of $g \in G$ on the Lie algebra $\mathfrak{g}$ is defined as the map $B_g : \mathfrak{g} \to \mathfrak{g}$ such that,*

$$B_g = \Psi^{-1} \circ L_g \circ \Psi,$$

with respect to the local coordinate map Ψ.

Note that if Ψ is the exponential map, such that $\Psi = \exp : \mathfrak{g} \to G$, then B_g is simply the Baker-Campbell-Hausdorff (BCH) formula $C(t)$, introduced in Section 1.3.7. For example, with respect to an element $\exp(x) \in G$, the map $B_{\exp(x)} : \mathfrak{g} \to \mathfrak{g}$ acting on an element y in $\mathfrak{g}$ yields,

$$B_{\exp(x)}(y) = (\exp^{-1} \circ L_{\exp(x)} \circ \exp)(y) = \exp^{-1} \circ (\exp(x) \cdot \exp(y)).$$

Comparing the above with Eq (1.43) shows that the mapping $B_{\exp(x)}$ is another way of writing the BCH formula.

Now, note that the composition of two equivariant maps is an equivariant map. So, we can split the idea of a single map from $\mathfrak{g}$ to $\mathcal{M}$, into a composition of two maps which are equivariant, namely for a point $x \in \mathcal{M}$, we have $\Psi : \mathfrak{g} \to G$, composed with the Lie group action $\Lambda(\cdot, x) : G \to \mathcal{M}$. Figure 3.2 demonstrates the commutativeness of such a composition map $\Lambda(\Psi(\cdot), x) : \mathfrak{g} \to \mathcal{M}$. This means that instead of studying the Lie group action $\Lambda(g, \cdot) : \mathcal{M} \to \mathcal{M}$ on the right hand side of Figure 3.2, the commutativeness of this diagram allows us to study the equivalent action of $B_g : \mathfrak{g} \to \mathfrak{g}$ on $\mathfrak{g}$.

Recall that a vector field is simply a map $F : \mathcal{M} \to T\mathcal{M}$, where $T\mathcal{M}$ is the tangent space of the manifold $\mathcal{M}$. In this case, we can consider the consequence of equivariance, which appears as relatedness (given by Eq (1.9)) between two vector fields $v_{\mathcal{M}}$ and $v_{\mathcal{N}}$. The two vector fields $v_{\mathcal{M}}$ and $v_{\mathcal{N}}$ are the infinitesimal generators of the Lie group action (or the flow) in Eq (1.32), with respect to the same element $v \in \mathfrak{g}$. A further discussion of the infinitesimal generators and

$$\begin{tikzcd}
\mathfrak{g} \arrow[r, "\Psi : \mathfrak{g} \to G"] & G \arrow[r, "\Lambda(\cdot, x) : G \to \mathcal{M}"] & \mathcal{M} \\
\mathfrak{g} \arrow[u, "B_g : \mathfrak{g} \to \mathfrak{g}"] \arrow[r, "\Psi : \mathfrak{g} \to G"'] & G \arrow[u, "L_g : G \to G"'] \arrow[r, "\Lambda(\cdot, x) : G \to \mathcal{M}"'] & \mathcal{M} \arrow[u, "\Lambda(g, \cdot) : \mathcal{M} \to \mathcal{M}"']
\end{tikzcd}$$

Figure 3.2: Equivariant map $\Lambda(\Psi(\cdot), x) : \mathfrak{g} \to \mathcal{M}$ with respect to the actions B_g on $\mathfrak{g}$ and Λ on $\mathcal{M}$, for all $g \in G$ and $x \in \mathcal{M}$.

its associated flow on $\mathcal{M}$ can be found in Section 1.3.3. Theorem 3.7 below provides the link between the idea of equivariance and relatedness. Note that the coordinate map $\Psi : \mathfrak{g} \to G$ used in the following discussion is the exponential map, and by Lemma 1.3, the Lie group action Λ is the flow (with respect to $v \in \mathfrak{g}$) of the infinitesimal generator $v_{\mathcal{M}}$.

Theorem 3.7. *Let $\phi : \mathcal{M} \to \mathcal{N}$ be a smooth map between two manifolds $\mathcal{M}$ and $\mathcal{N}$. Also, let $\Lambda_1(\exp(tv), \cdot) : \mathcal{M} \to \mathcal{M}$ be the flow of the infinitesimal generator $v_{\mathcal{M}}$, and $\Lambda_2(\exp(tv), \cdot) : \mathcal{N} \to \mathcal{N}$ be the flow of $v_{\mathcal{N}}$. Then for a point $p \in \mathcal{M}$, ϕ is an equivariant map,*

$$\phi\left(\Lambda_1(\exp(tv), p)\right) = \Lambda_2(\exp(tv), \phi(p)), \qquad \forall t \in \mathbb{R}, v \in \mathfrak{g},$$

if and only if the vector fields $v_{\mathcal{M}}$ and $v_{\mathcal{N}}$ are ϕ-related ($v_{\mathcal{M}} \sim_\phi v_{\mathcal{N}}$),

$$\phi' \circ v_{\mathcal{M}} = v_{\mathcal{N}} \circ \phi.$$

Proof. To go from the equivariant relationship to ϕ-relatedness, start by differentiating the equivariant relationship with respect to t.

$$\begin{aligned}
\frac{d}{dt}\left(\phi \circ \Lambda_1(\exp(tv), p)\right) &= \frac{d}{dt}\Lambda_2(\exp(tv), \phi(p)) \\
\Rightarrow \phi' \circ \frac{d}{dt}\Lambda_1(\exp(tv), p) &= \frac{d}{dt}\Lambda_2(\exp(tv), \phi(p)).
\end{aligned}$$

Since Λ_1 is the flow of the infinitesimal generator $v_{\mathcal{M}}$, and Λ_2 is the flow of $v_{\mathcal{N}}$, by Lemma 1.3 in Section 1.3.3,

$$\left(\phi' \circ v_{\mathcal{M}}\right)\left(\Lambda_1(\exp(tv), p)\right) = v_{\mathcal{N}}\left(\Lambda_2(\exp(tv), \phi(p))\right).$$

Rewriting this in composition notation on the left hand side, and noting that ϕ is an equivariant map between the flows Λ_1 and Λ_2, we can derive the desired ϕ-relatedness ($v_{\mathcal{M}} \sim_\phi v_{\mathcal{N}}$) between the infinitesimal generators.

$$\begin{aligned}
(\phi' \circ v_{\mathcal{M}} \circ \Lambda_1)(\exp(tv), p) &= (v_{\mathcal{N}} \circ \phi \circ \Lambda_1)(\exp(tv), p) \\
\phi' \circ v_{\mathcal{M}} &= v_{\mathcal{N}} \circ \phi.
\end{aligned}$$

Now, to go from ϕ-relatedness back to the equivariance relationship, we use the definition of the integral curve. Let the flow at point $p \in \mathcal{M}$ be denoted $\Lambda_1(\exp(tv), p)$ such that as $t \in \mathbb{R}$ varies, it is an integral curve through $p \in \mathcal{M}$ (i.e. at $t = 0$, $\Lambda_1(\exp(tv), p) = p$), with infinitesimal generator $v_{\mathcal{M}}$. We then have an equation similar to Eq (1.32) in Corollary 1.1,

$$\frac{d}{dt}\Lambda_1(\exp(tv), p) = v_{\mathcal{M}}(\Lambda_1(\exp(tv), p)).$$

Then, we can differentiate $\phi(\Lambda_1(\exp(tv),p))$ with respect to t, and use the ϕ-relatedness between $v_{\mathcal{M}}$ and $v_{\mathcal{N}}$ to derive the following,

$$\begin{aligned} \frac{d}{dt}\left(\phi(\Lambda_1(\exp(tv),p))\right) &= \phi' \circ \frac{d}{dt}\Lambda_1(\exp(tv),p) \\ &= \phi' \circ v_{\mathcal{M}}\left(\Lambda_1(\exp(tv),p)\right) \\ &= v_{\mathcal{N}} \circ \phi\left(\Lambda_1(\exp(tv),p)\right). \end{aligned}$$

Note that the above calculation shows that the flow $\phi(\Lambda_1(\exp(tv),p))$ is an integral curve of the infinitesimal generator $v_{\mathcal{N}}$ through the point $\phi(\Lambda_1(\exp(0),p)) = \phi(p)$ on the manifold $\mathcal{N}$ at $t=0$. By uniqueness, this integral curve on $\mathcal{N}$ must be the same as the integral curve $\Lambda_2(\exp(tv),\phi(p))$ with respect to the point $\phi(p) \in \mathcal{N}$ at $t=0$. Hence, we have the equivariant relationship,

$$\phi(\Lambda_1(\exp(tv),p)) = \Lambda_2(\exp(tv),\phi(p)), \qquad \forall t \in \mathbb{R}, v \in \mathfrak{g}, p \in \mathcal{M}.$$

□

In a more abstract setting, where $\Psi : \mathfrak{g} \to G$ is the coordinate map and $p \in \mathcal{M}$, then by the equivariant map $\Lambda(\Psi(\cdot),p) : \mathfrak{g} \to \mathcal{M}$, we have the following commutative diagram in Figure 3.3 for $v \in \mathfrak{g}$.

$$\begin{tikzcd}[column sep=huge, row sep=large]
\mathfrak{g} \arrow[r, "\Psi : \mathfrak{g} \to G"] & G \arrow[r, "\Lambda(\cdot,p) : G \to \mathcal{M}"] & \mathcal{M} \\
\mathfrak{g} \arrow[u, "B_{\exp(tv)} : \mathfrak{g} \to \mathfrak{g}"] \arrow[r, "\Psi : \mathfrak{g} \to G"'] & G \arrow[u, "L_{\exp(tv)} : G \to G"'] \arrow[r, "\Lambda(\cdot,p) : G \to \mathcal{M}"'] & \mathcal{M} \arrow[u, "\Lambda(\exp(tv),\cdot) : \mathcal{M} \to \mathcal{M}"']
\end{tikzcd}$$

Figure 3.3: Equivariant map $\Lambda(\Psi(\cdot),p) : \mathfrak{g} \to \mathcal{M}$ with respect to the element $\exp(tv) \in G$, where $v \in \mathfrak{g}$, for all $t \in \mathbb{R}$ and $p \in \mathcal{M}$.

From Theorem 3.7, we know that $\Lambda(\Psi(\cdot),p) : \mathfrak{g} \to \mathcal{M}$ is an equivariant map on the flows $B_{\exp(tv)} : \mathfrak{g} \to \mathfrak{g}$, and $\Lambda(\exp(tv),\cdot) : \mathcal{M} \to \mathcal{M}$, if and only if the infinitesimal generators $v_{\mathcal{M}}$ and $v_{\mathfrak{g}}$ are $(\Lambda \circ \Psi)$-related. That is,

$$(\Lambda \circ \Psi)' \circ v_{\mathfrak{g}} = v_{\mathcal{M}} \circ (\Lambda \circ \Psi).$$

The infinitesimal generator $v_{\mathcal{M}}$ is defined by Eq (1.32) in Corollary 1.1. To find out what the infinitesimal generator $v_{\mathfrak{g}}$ is, we can use the following theorem.

Theorem 3.8. *Let $\Psi : \mathfrak{g} \to G$. Suppose that for $v \in \mathfrak{g}$, $\Lambda(\Psi(\cdot),p) : \mathfrak{g} \to \mathcal{M}$ is an equivariant map with respect to the flows $B_{\exp(tv)} : \mathfrak{g} \to \mathfrak{g}$, and $\Lambda(\exp(tv),\cdot) : \mathcal{M} \to \mathcal{M}$. Then the infinitesimal generator of $B_{\exp(tv)}$ satisfying the relationship $(\Lambda \circ \Psi)' \circ v_{\mathfrak{g}} = v_{\mathcal{M}} \circ (\Lambda \circ \Psi)$, is given by,*

$$v_{\mathfrak{g}}(u) = \mathrm{d}\Psi_u^{-1}(v), \qquad u \in \mathfrak{g},$$

where $\mathrm{d}\Psi : \mathfrak{g} \to \mathfrak{g}$ is the trivialisation of the derivative Ψ',

$$\mathrm{d}\Psi_u = R'_{\Psi(u)^{-1}} \circ \Psi'_u.$$

Proof. From the right hand side of the commutative diagram in Figure 3.3, we can consider the Lie group action Λ as an equivalent map such that we have the relationship for $t \in \mathbb{R}$, and $v \in \mathfrak{g}$,

$$\Lambda(L_{\exp(tv)}(g), p) = \Lambda(\exp(tv), \Lambda(g, p)), \qquad \forall g \in G, \text{ and } p \in \mathcal{M}.$$

By Theorem 3.7, since Λ is an equivariant map, then the infinitesimal generators v_G and $v_{\mathcal{M}}$ are Λ-related. So we have the commutative diagram in Figure 3.4. Note that $R'_{\cdot}(v) : G \to TG$ is the derivative of the 'right' multiplication. The Λ-relatedness between the infinitesimal generators v_G and $v_{\mathcal{M}}$ is given by, for $v \in \mathfrak{g}$, $t \in \mathbb{R}$, and a point $p \in \mathcal{M}$,

$$(v_{\mathcal{M}} \circ \Lambda)(\exp(tv), p) = \frac{d}{dt}\Lambda(R_p(\exp(tv))) = (\Lambda' \circ R'_p)(v).$$

$$\begin{tikzcd}
TG \arrow[r, "\Lambda'(\cdot,p) : TG \to T\mathcal{M}"] & T\mathcal{M} \\
G \arrow[u, "v_G = R'_{\cdot}(v) : G \to TG"] \arrow[r, "\Lambda(\cdot,p) : G \to \mathcal{M}"'] & \mathcal{M} \arrow[u, "v_{\mathcal{M}} : \mathcal{M} \to T\mathcal{M}"']
\end{tikzcd}$$

Figure 3.4: Λ-relatedness between infinitesimal generators v_G and $v_{\mathcal{M}}$.

Now, since Λ is an equivalence map between G and $\mathcal{M}$, then instead of studying the $\Lambda(\Psi(\cdot), p) : \mathfrak{g} \to \mathcal{M}$ equivariant mapping between $\mathfrak{g}$ and $\mathcal{M}$ in Figure 3.3, we can simply look at the Ψ-equivariance between $\mathfrak{g}$ and G, with respect to the flows $B_{\exp(tv)} : \mathfrak{g} \to \mathfrak{g}$ and $L_{\exp(tv)} : G \to G$. That is, for $u \in \mathfrak{g}$ we have,

$$\Psi \circ B_{\exp(tv)}(u) = L_{\exp(tv)} \circ \Psi(u).$$

We will restrict ourselves to the left hand side of the commutative diagram in Figure 3.3. The derivatives of this commutative diagram (or the tangent lifts in $\mathfrak{g}$ and G), are shown in Figure 3.5. By Theorem 3.7, since Ψ is an equivariant map, we now have Ψ-relatedness between the infinitesimal generators $v_{\mathfrak{g}}$ and v_G, such that for $u \in \mathfrak{g}$ in Figure 3.5,

$$(\Psi' \circ v_{\mathfrak{g}})(u) = (v_G \circ \Psi)(u).$$

where, in Figure 3.5, the notation $\Psi' : \mathfrak{g} \to TG$ denotes the **tangent lift** of the mapping $\Psi : \mathfrak{g} \to G$.

$$\begin{tikzcd}
T\mathfrak{g} = \mathfrak{g} \arrow[r, "\Psi' : \mathfrak{g} \to TG"] & TG \\
\mathfrak{g} \arrow[u, "v_{\mathfrak{g}} : \mathfrak{g} \to T\mathfrak{g} = \mathfrak{g}"] \arrow[r, "\Psi : \mathfrak{g} \to G"'] & G \arrow[u, "v_G = R'_{\cdot}(v) : G \to TG"']
\end{tikzcd}$$

Figure 3.5: Ψ-relatedness between infinitesimal generators $v_{\mathfrak{g}}$ and v_G.

To proceed, recall that the infinitesimal generator v_G is given by the derivative of the 'Right multiplication' map, such that,

$$v_G = R'_{\cdot}(v) : G \to TG, \qquad \text{for } v \in \mathfrak{g}.$$

and that the derivative of the map $\Psi : \mathfrak{g} \to G$ is itself a composition of two maps, $\mathrm{d}\Psi_u : \mathfrak{g} \to \mathfrak{g}$, and $R'_{\Psi(u)} : \mathfrak{g} \to TG$, such that we have,

$$\Psi' = R'_{\Psi(u)} \circ \mathrm{d}\Psi_u, \qquad \text{for } u \in \mathfrak{g}.$$

Hence, for Ψ-relatedness between two vector fields $v_{\mathfrak{g}}$ and v_G we have, for $u, v \in \mathfrak{g}$,

$$\begin{aligned} (\Psi' \circ v_{\mathfrak{g}})(u) &= (v_G \circ \Psi)(u) \\ R'_{\Psi(u)} \circ d\Psi_u \circ v_{\mathfrak{g}}(u) &= R'_{\Psi(u)}(v). \end{aligned}$$

The above is true if we have,

$$v_{\mathfrak{g}}(u) = d\Psi_u^{-1}(v).$$

□

3.2 The RKMK Class

This section is a summary of what happens if we apply the theory in the previous section to a numerical algorithm. The main idea behind the RKMK class of integrators is to rewrite a differential equation evolving on $\mathcal{M}$, into one that evolves on $\mathfrak{g}$. This is because $\mathfrak{g}$ has the structure of a vector space, and the classical Runge-Kutta (RK) Method can be applied to numerically solve this equivalent differential equation on $\mathfrak{g}$. To do this, we construct an equivariant map $\Lambda(\Psi(\cdot), p) : \mathfrak{g} \to \mathcal{M}$ for $p \in \mathcal{M}$, as in Figure 3.3. Also, since $\mathfrak{g}$ is the Lie algebra associated with the Lie group G acting on $\mathcal{M}$, then we can find the elements of $v \in \mathfrak{g}$, such that we can construct the infinitesimal generators (defined by Eq (1.32)), which describe the original differential equation in terms of the Lie group action $\Lambda : G \times \mathcal{M} \to \mathcal{M}$.

In other words, we want to solve the differential equation on $\mathcal{M}$,

$$y' = F(y), \qquad t \geq 0,\ y(0) = y_0 \in \mathcal{M},$$

where at each point $y(t) \in \mathcal{M}$ for $t \in \mathbb{R}$, the vector field $F(y(t))$ can be written in terms of the infinitesimal generator $v_{\mathcal{M}}$,

$$F(y) = v_{\mathcal{M}}(y) = \frac{d}{dt}\Lambda(\exp(tv), y),$$

for some $v \in \mathfrak{g}$ which relates to the differential equation on $\mathcal{M}$. Once v is found, the transformation to an equivalent system on $\mathfrak{g}$ is done by solving the differential equation given below (for the coordinate map $\Psi : \mathfrak{g} \to G$, and $v \in \mathfrak{g}$ with respect to the infinitesimal generator $v_{\mathcal{M}}$),

$$u' = \mathrm{d}\Psi_u^{-1}(v), \qquad u(0) = u_0.$$

Hence, $u(t)$, for $t \in \mathbb{R}$, is the solution of the equivalent differential equation evolving on $\mathfrak{g}$. Once $u(t)$ is approximated using the classical integrators like the Runge-Kutta method, then the corresponding solution $y(t) \in \mathcal{M}$ can be found by the Lie group action, with respect to the known point, say $y_0 \in \mathcal{M}$,

$$y(t) = \Lambda(\Psi(u(t)), y_0).$$

Note that very often, to relate the element $v \in \mathfrak{g}$ to the original differential equation on $\mathcal{M}$, v is defined as the map $f : \mathbb{R} \times \mathcal{M} \to \mathfrak{g}$, with respect to some previously determined point like $y(t)$ on $\mathcal{M}$, given by the Lie group action $\Lambda : G \times \mathcal{M} \to \mathcal{M}$ [28]. That is,

$$v(t) = f(t, \Lambda(\Psi(u(t)), y(t))) \qquad \text{for } t \in \mathbb{R}.$$

This kind of mapping for $v \in \mathfrak{g}$ is referred to as the **generator map**, such that the original differential equation can be written as an infinitesimal generator $v_{\mathcal{M}}$ with respect to the action Λ and the mapping $f : \mathbb{R} \times \mathcal{M} \to \mathfrak{g}$, which is often given by the differential equation [12].

Hence, we can solve the differential equation evolving on the homogeneous space $\mathcal{M}$ that can be described as an infinitesimal generator $v_{\mathcal{M}}$. When presenting this theory as an algorithm, the following steps should be taken:

Overall Algorithm

Assume that the vector field $F \in \mathfrak{X}(\mathcal{M})$ over the homogeneous space $\mathcal{M}$ can be written as an infinitesimal generator $v_{\mathcal{M}}$ of the action $\Lambda : G \times \mathcal{M} \to \mathcal{M}$. (i.e. $F = v_{\mathcal{M}}$ for some $v \in \mathfrak{g}$ which describes the original differential equation.) The following structure is designed for using one step of a qth order classical Runge-Kutta method to approximate the solution y from $t_0 = 0$ to $t_1 > 0$ of the original differential equation [11].

1. For the original differential equation,

$$y' = F(y), \qquad t \geq 0, \quad y(0) = y_0 \in \mathcal{M},$$

   determine the manifold $\mathcal{M}$, Lie group G and Lie algebra $\mathfrak{g}$. Next, find the Lie group action $\Lambda : G \times \mathcal{M} \to \mathcal{M}$, and the mapping $f : \mathbb{R} \times \mathcal{M} \to \mathfrak{g}$ such that the differential equation can be written as an infinitesimal generator $v_{\mathcal{M}}$.

2. Choose the initial point $u_0 = 0$ in $\mathfrak{g}$. For any initial point $y_0 \in \mathcal{M}$ we have $y_0 = \Lambda(\Psi(u_0), y_0)$, where $\Psi : \mathfrak{g} \to G$.

3. Solve the equivalent differential equation on $\mathfrak{g}$, using any qth order Runge-Kutta method. This equivalent differential equation is given by,

$$u' = \mathrm{d}\Psi_u^{-1}(f(t, y)) = \mathrm{d}\Psi_u^{-1}(f(t, \Lambda(\Psi(u), y_0)), \qquad u(0) = u_0.$$

   Denote this approximation to the solution $u(t_1)$ by say, u_1.

4. Find the corresponding approximation to the solution $y_1 \approx y(t_1)$, by the equivariant map $\Lambda(\Psi(\cdot), y_0) : \mathfrak{g} \to \mathcal{M}$, with respect to the initial point $y_0 \in \mathcal{M}$. That is, the $y(t_1)$ approximate is given by $y_1 = \Lambda(\Psi(u_1), y_0)$. This completes one step of this algorithm, and we now have an approximation to $y(t_1) \in \mathcal{M}$ at $t_1 > 0$.

3.3 Canonical Coordinates of the First Kind

In this section, we will look at the different choices of the local coordinate maps $\Psi : \mathfrak{g} \to G$, which can be used to form the differential equation for $u(t)$ evolving in the linear space $\mathfrak{g}$. In particular, we will look at the exponential map and the Cayley map for the implementation of the RKMK method, and how a qth order Runge-Kutta method is incorporated into the RKMK scheme explicitly. The use of the exponential map exp and the Cayley map cay are sometimes referred to as the **Canonical Coordinates of the first kind**.

3.3.1 The Exponential Map

The exponential map $\Psi = \exp : \mathfrak{g} \to G$ was used by Munthe-Kaas in [28], when the RKMK scheme was presented in its final form. We will first look at the algorithm, then follow the detail in [28].

Given that $u, v \in \mathfrak{g}$, and q is the order of the Runge-Kutta method, the truncated expression of the inverse of the differential of the exponential map in Eq (1.35), Theorem 1.3 is given by,

$$\texttt{dexpinv}(u, v, q) = \sum_{k=0}^{q-2} \frac{B_k}{k!} \mathrm{ad}_u^k(v).$$

This truncated expression will be used in the RKMK Algorithm, as quoted from [28], and a discussion for truncating this series at $q-2$ is given at the end of this section.

The RKMK Algorithm
From $t_0 = 0$ to $t_f = h$, one step of the RKMK Algorithm is given by the following scheme :

> Let $a_{i,j}$ and b_j be the coefficients of an s-stage, q'th order classical Runge-Kutta method and let $c_i = \sum_{j=1}^{s} a_{i,j}$,
> $y_0 = p$
> for $i = 1, 2, ..., s$
> $u_i = h \sum_{j=1}^{s} a_{i,j} \widetilde{k}_j$
> $k_i = f(hc_i, \lambda(u_i, y_0))$
> $\widetilde{k}_i = \mathrm{dexpinv}(u_i, k_i, q)$
> end
> $v = h \sum_{j=1}^{s} b_j \widetilde{k}_j$
> $y_1 = \lambda(v, y_0)$

Note that $f : \mathbb{R} \times \mathcal{M} \to \mathfrak{g}$, and the Lie algebra action is $\lambda : \mathfrak{g} \times \mathcal{M} \to \mathcal{M}$.

Associated with this algorithm, Munthe-Kaas also presented the following theorems and lemma in [28], which we will discuss in order to get a better understanding of the structure behind the RKMK Algorithm.

Theorem 3.9. *The RKMK Algorithm stays on the manifold $\mathcal{M}_p \subset \mathcal{M}$ for $p \in \mathcal{M}$,*

$$\mathcal{M}_p = \{q \in \mathcal{M} \mid \exists v_i \in \mathfrak{g},\ \textit{such that}\ q = \lambda(v_k, \ldots, \lambda(v_2, \lambda(v_1, p)))\}.$$

Proof. This follows from the observation that all approximations take place in the Lie algebra $\mathfrak{g}$, and that the Lie algebra action $\lambda : \mathfrak{g} \times \mathcal{M} \to \mathcal{M}$. □

The idea behind Munthe-Kaas's RKMK presentation is similar to the derivation for the equivalent differential equation for $u(t)$ given earlier in Section 3.2. In this case, Munthe-Kaas proved that if $u' \in \mathfrak{X}(\mathfrak{g})$ based on the exponential map is given by the vector field, where $f : \mathcal{M} \to \mathfrak{g}$, and $\lambda : \mathfrak{g} \times \mathcal{M} \to \mathfrak{g}$,

$$u'(t) = \mathrm{dexp}_u^{-1}(f \circ \lambda(u, p)), \qquad \text{for } t \in \mathbb{R},\ p \in \mathcal{M}, \text{ and } u \in \mathfrak{g},$$

then u' must be λ-related to the original vector field $y' = F(y)$ evolving on $\mathcal{M}$. Note that in this notation, the Lie algebra action λ, is the composition of two equivariant maps $\Lambda \circ \Psi$, where $\Lambda : G \times \mathcal{M} \to \mathcal{M}$, and $\Psi = \exp : \mathfrak{g} \to G$. This commutivity can be seen in the Figure 3.6, very similar to Figure 3.4 and Figure 3.5 in Section 3.1.

The details are given in the following Lemma from [28].

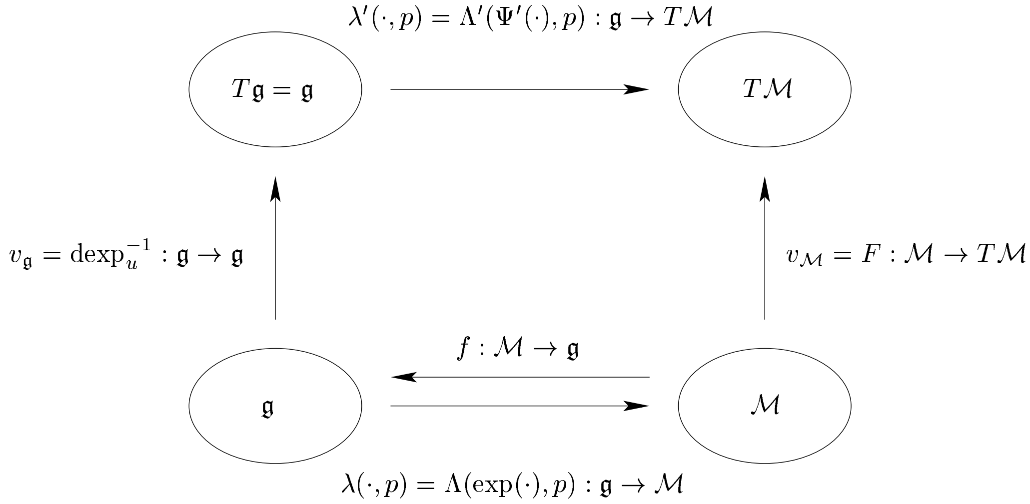


Figure 3.6: λ-relatedness between $u' = \mathrm{dexp}_u^{-1}(v)$ and $y' = F(y)$.

Lemma 3.9. *Let $F \in \mathfrak{X}(\mathcal{M})$ at the point $p \in \mathcal{M}$ be the vector field $F(p) = (\lambda_* f(p))(p)$ for some $f : \mathcal{M} \to \mathfrak{g}$. If $u' \in \mathfrak{X}(\mathfrak{g})$ is the vector field given by,*

$$u' = \mathrm{dexp}_u^{-1}(f \circ \lambda(u,p)), \qquad \textit{for } p \in \mathcal{M}, \tag{3.1}$$

then $u' \sim_\lambda F$.

Proof. To begin this proof, recall the definition of the local group action $\Lambda : G \times \mathcal{M} \to \mathcal{M}$ is related to the Lie algebra action $\lambda : \mathfrak{g} \times \mathcal{M} \to \mathcal{M}$ from Eq (1.31),

$$\lambda(u,p) = \Lambda(\exp(u),p), \qquad \text{for } u \in \mathfrak{g}, \text{ and } p \in \mathcal{M},$$

and the derivative of the exponential map in Theorem 1.3 is given by,

$$\exp(u+tv)'\big|_{t=0} = R'_{\exp(u)} \circ \mathrm{dexp}_u(v) \qquad \text{for } u, v \in \mathfrak{g}.$$

For u' to be λ-related to F ($u' \sim_\lambda F$), we need to prove the following relationship,

$$\lambda'(u',p) = F \circ \lambda(u,p). \tag{3.2}$$

Starting from the left hand side of Eq (3.2), we have the following, for $u \in \mathfrak{g}$ and $p \in \mathcal{M}$,

$$\begin{aligned}
\lambda'(u',p) &= \Lambda'(\exp(u)',p) && {}' \text{ denotes time derivative.}\\
&= \Lambda'(R'_{\exp(u)} \circ \mathrm{dexp}_u(u'),p) && \text{derivative of exp.}\\
&= \Lambda'(R'_{\exp(u)} \circ \mathrm{dexp}_u(\mathrm{dexp}_u^{-1}(f \circ \lambda(u,p))),p) && \text{substitute definition of } u'.\\
&= \lambda'(R'_{\exp(u)}(f \circ \lambda(u,p)),p).
\end{aligned}$$

Starting from the right hand side of Eq (3.2), we can rewrite it in the form of the generic presentation of differential equation in Assumption 1.1 in Section 1.3.5,

$$F \circ \lambda(u,p) = (\lambda_* f(\lambda(u,p)))\,(\lambda(u,p)) = (\lambda_* v)\,(\lambda(u,p)), \tag{3.3}$$

where $F \in \mathfrak{X}(\mathcal{M})$, $\lambda : \mathfrak{g} \times \mathcal{M} \to \mathcal{M}$, $f : \mathcal{M} \to \mathfrak{g}$, and we make the substitution $v = f \circ \lambda(u,p) \in \mathfrak{g}$.

Recall that each $v \in \mathfrak{g}$ generates a vector field on $\mathcal{M}$ via the infinitesimal generator given in Corollary 1.1 (which is the same as the 'generic presentation' in Assumption 1.1),

$$v_{\mathcal{M}} = (\lambda_* v)\,(p) = \frac{d}{dt}\lambda(tv,p)\bigg|_{t=0} = \frac{d}{dt}\Lambda(\exp(tv),p)\bigg|_{t=0} \qquad \text{for } t \in \mathbb{R},\ p \in \mathcal{M}.$$

Hence, we can rewrite the right hand side of Eq (3.3) as,

$$\begin{aligned}
(\lambda_* v)\,(\lambda(u,p)) &= \lambda(tv,\lambda(u,p))'\big|_{t=0} && {}' \text{ denotes time derivative.}\\
&= \Lambda(\exp(tv),\Lambda(\exp(u),p))'\big|_{t=0} && \\
&= \Lambda(\exp(tv)\cdot\exp(u),p)'\big|_{t=0} && \text{by local Lie group action.}\\
&= \Lambda(R_{\exp(u)}\circ\exp(tv),p)'\big|_{t=0} && \text{by right multiplication.}\\
&= \Lambda'(R'_{\exp(u)}\circ\exp(tv)',p)\big|_{t=0} && \\
&= \Lambda'(R'_{\exp(u)}(v),p) && \exp(tv)'|_{t=0} = v.\\
&= \Lambda'(R'_{\exp(u)}(f\circ\lambda(u,p)),p) && \text{substitute } v = f\circ\lambda(u,p).
\end{aligned}$$

Hence, we can see that both the left and right hand sides of Eq (3.2) are equal, and so $u' \sim_{\lambda_p} F$, which completes the proof of the above Lemma. $\square$

Given that we now have an expression for the vector field u', which is λ-related to the original vector field $y' = F(y)$, we can then put everything together in the following corollary from [28].

Corollary 3.2. *The solution to the generic presentation of $y' = v_{\mathcal{M}}$ is given, for sufficiently small t, as,*

$$y(t) = \lambda(u(t),p),$$

where $u(t) \in \mathfrak{g}$, for $t \in \mathbb{R}$, satisfy the differential equation

$$u' = \mathrm{dexp}_u^{-1}(f(t,\lambda(u,p))), \qquad u(0) = 0. \tag{3.4}$$

Again, we arrive at the idea discussed previously in Section 3.2, where we want to write the differential equation in its generic form with infinitesimal generators $v_{\mathcal{M}}$. Then we can find an equivalent differential equation for u, which is given by the inverse of the differential dexp_u^{-1}. This is possible because the two vector fields F and dexp_u^{-1} are λ-related. Since u' evolves on the linear vector space $\mathfrak{g}$, then traditional methods like the Runge-Kutta methods can be used to approximate the solution for $u(t) \in \mathfrak{g}$ at some time t. Once $u(t)$ is found, we can then find the equivalent $y(t)$ by using the Lie group (algebra) action given in Corollary 3.2.

To finish this section, we now discuss one last theorem presented in [28], which is related to the order of the RKMK method.

Theorem 3.10. *The RKMK Algorithm has at least order q for any Lie algebra action λ on any manifold $\mathcal{M}$, if the RK Algorithm involved is of order q.*

Proof. The RKMK method has order q if we can match the first $q+1$ terms in the Lie series of the numerical solution $y_1(h)$ at $h = 0$, with the first $q+1$ terms in the Lie series from the analytical solution of Eq (1.39) in Assumption 1.1 at $t = 0$. This can be done by using the Lie-Butcher series developed by Munthe-Kaas in his earlier work. However, the theorem can also be proved without the use of the Lie-Butcher series. Note that RKMK Algorithm is equivalent to doing a single classical RK step in the Lie algebra $\mathfrak{g}$ on the differential equation in Eq (3.4), then transferring back to the manifold $\mathcal{M}$ via the Lie algebra action $y_1 = \lambda(u_1, y_0)$. Since $\mathfrak{g}$ is a vector space, and λ is a smooth mapping, then the order of RKMK on $\mathcal{M}$ is at least as high as the order of the corresponding equation on $\mathfrak{g}$.

Also, in numerical computation of Eq (3.4), u' is given by dexp_u^{-1}, which is truncated at the $k = (q-2)$th term. This is because all Bernoulli's coefficients for odd $k \geq 3$ are zero, so that for an error estimate of dexp_u^{-1} of order $O(h^{q+1})$, only terms up to $k = q-2$ are needed. This can be seen more clearly if we write out the terms in the expansion, such that,

$$\mathrm{dexp}_u^{-1}(v) = v - \frac{1}{2}\mathrm{ad}_u^1(v) + \frac{1}{12}\mathrm{ad}_u^2(v) + 0 - \frac{1}{720}\mathrm{ad}_u^4(v) + 0 - \cdots$$

where the first term v is of the order $O(h)$, second term $\mathrm{ad}_u^1(v)$ is of the order $O(h^2)$, third term $\mathrm{ad}_u^2(v)$ is $O(h^3)$, fifth term $\mathrm{ad}_u^4(v)$ is $O(h^5)$, and so on. This implies that for a numerical method of order $q = 4$, if we want to preserve this order when substituting into Eq (3.4), we only need the expansion up to the third term, when $k = 2$. Hence the truncation in dexp_u^{-1} in the RKMK Algorithm does not reduce the order of the numerical scheme. □

3.3.2 The Cayley Map

If the Lie group under consideration is a quadratic Lie group, then we can replace the exponential map, with the Cayley transform $\mathrm{cay} : \mathfrak{g} \to G$ [17].

The quadratic Lie group G_M, where $M \in \mathrm{GL}(n)$, is defined as [8],

$$G_M = \{g \in \mathrm{GL}(n) : g^T M g = M\},$$

and the quadratic Lie algebra is given by,

$$\mathfrak{g}_M = \{u \in \mathfrak{gl}(n) : u^T M + M u = 0\}.$$

Note that the orthogonal Lie group O(n), the special orthogonal Lie group $\mathrm{SO}(n)$, and the symplectic Lie group Sp(n) are quadratic Lie groups. This can be seen if the matrix M is defined to be $M = \mathrm{Id}$ (the identity matrix) for the former, and $M = J$ for even dimensions for the symplectic group, where,

$$J = \begin{bmatrix} 0 & \mathrm{Id} \\ -\mathrm{Id} & 0 \end{bmatrix}.$$

Any map $r(z)$ which is analytic at 0 and such that $r(z)r(-z) = 1$ will be able to map the quadratic Lie algebra $\mathfrak{g}_M$ to its corresponding quadratic Lie group G_M [8]. This includes the diagonal Padé approximants [19], and one such example is the function $r(z) = \frac{(1+z)}{(1-z)}$. In [17], the Cayley transform is in fact defined as the following,

$$\mathrm{cay}(z, \alpha) = \frac{(1+\alpha z)}{(1-\alpha z)}, \quad \alpha \in \mathbb{R} \setminus \{0\}.$$

The advantage of using the Cayley transform as the coordinate map $\Psi : \mathfrak{g} \to G$ is the relatively low cost in its computation as compared with the exponential map. Note that if $\alpha = \frac{1}{2}$, then the Cayley transform is simply the Padé Approximation to the exponential map, denoted $R(1,1)$, where for two matrices u and v in the quadratic Lie algebra $\mathfrak{g}_M$,

$$\mathrm{cay}(u) = \left(\mathrm{Id} - \frac{u}{2}\right)^{-1}\left(\mathrm{Id} + \frac{u}{2}\right). \tag{3.5}$$

Similarly to the differential of the exponential map in Theorem 1.3 in Chapter 1, the differential dcay_u and its inverse dcay_u^{-1} are presented in the following lemma.

Lemma 3.10. *For $u, v \in \mathfrak{g}_M$, the derivative of the Cayley transform is given by,*

$$\frac{d}{dt}\operatorname{cay}(u+tv)\big|_{t=0} = R'_{\operatorname{cay}_u} \circ \operatorname{dcay}_u(v) = \operatorname{dcay}_u(v)\cdot\operatorname{cay}(u),$$

where $\operatorname{dcay}_u(v)$ *and* $\operatorname{dcay}_u^{-1}(v)$ *are the differential of the Cayley transform and its inverse map,*

$$\operatorname{dcay}_u(v) = \left(\mathrm{Id}-\frac{u}{2}\right)^{-1} v \left(\mathrm{Id}+\frac{u}{2}\right)^{-1} \tag{3.6}$$

$$\operatorname{dcay}_u^{-1}(v) = \left(\mathrm{Id}-\frac{u}{2}\right) v \left(\mathrm{Id}+\frac{u}{2}\right) = v - \frac{1}{2}\mathrm{ad}_u(v) - \frac{1}{4}uvu. \tag{3.7}$$

Proof. Let us assume that the derivative of the Cayley transform has the form given above for $u, v \in \mathfrak{g}_M$. We can find an expression for $\operatorname{dcay}_u(v)$ by using the following,

$$\mathrm{Id}+\operatorname{cay}(u) = \left(\mathrm{Id}-\frac{u}{2}\right)^{-1} 2\,\mathrm{Id}. \tag{3.8}$$

To prove Eq (3.8), we use the definition of the Cayley transform in Eq (3.5),

$$\begin{aligned}\mathrm{Id}+\operatorname{cay}(u) &= \mathrm{Id}+\left(\mathrm{Id}-\frac{u}{2}\right)^{-1}\left(\mathrm{Id}+\frac{u}{2}\right) = \left(\mathrm{Id}-\frac{u}{2}\right)^{-1}\left(\left(\mathrm{Id}-\frac{u}{2}\right)+\left(\mathrm{Id}+\frac{u}{2}\right)\right)\\ &= \left(\mathrm{Id}-\frac{u}{2}\right)^{-1} 2\,\mathrm{Id}.\end{aligned}$$

For the Cayley transform of $\operatorname{cay}(u+tv)$ where $t \in \mathbb{R}$, we again use the definition in Eq (3.5) and rearrange to find,

$$\left(\mathrm{Id}-\frac{u}{2}-\frac{tv}{2}\right)\operatorname{cay}(u+tv) = \left(\mathrm{Id}+\frac{u}{2}+\frac{tv}{2}\right). \tag{3.9}$$

Expanding $\operatorname{cay}(u+tv)$ using Taylors series expansion,

$$\begin{aligned}\operatorname{cay}(u+tv) &= \operatorname{cay}(u) + t\left[\frac{d}{dt}\operatorname{cay}(u+tv)\right]\Big|_{t=0} + O(t^2)\\ &= \operatorname{cay}(u) + t\operatorname{dcay}_u(v)\cdot\operatorname{cay}(u) + O(t^2).\end{aligned}$$

So, substituting the above in Eq (3.9), we find,

$$\left(\mathrm{Id}-\frac{u}{2}-\frac{tv}{2}\right)\left(\operatorname{cay}(u)+t\operatorname{dcay}_u(v)\cdot\operatorname{cay}(u)+O(t^2)\right) = \left(\mathrm{Id}+\frac{u}{2}+\frac{tv}{2}\right).$$

Differentiating this expression with respect to t, and then set $t=0$ yields,

$$\begin{aligned}-\frac{v}{2}\operatorname{cay}(u) + \left(\mathrm{Id}-\frac{u}{2}\right)\Big(\operatorname{dcay}_u(v)\operatorname{cay}(u)\Big) &= \frac{v}{2}\\ \left(\mathrm{Id}-\frac{u}{2}\right)\Big(\operatorname{dcay}_u(v)\operatorname{cay}(u)\Big) &= \frac{v}{2}\Big(\mathrm{Id}+\operatorname{cay}(u)\Big).\end{aligned}$$

Substituting Eq (3.8), and the definition of the Cayley transform into the above, we find,

$$\begin{aligned}\left(\mathrm{Id}-\frac{u}{2}\right)\operatorname{dcay}_u(v)\left(\mathrm{Id}-\frac{u}{2}\right)^{-1}\left(\mathrm{Id}+\frac{u}{2}\right) &= v\left(\mathrm{Id}-\frac{u}{2}\right)^{-1}\\ \left(\mathrm{Id}-\frac{u}{2}\right)\operatorname{dcay}_u(v)\left(\mathrm{Id}-\frac{u}{2}\right)^{-1}\left(\mathrm{Id}+\frac{u}{2}\right)\left(\mathrm{Id}-\frac{u}{2}\right) &= v\\ \left(\mathrm{Id}-\frac{u}{2}\right)\operatorname{dcay}_u(v)\left(\mathrm{Id}-\frac{u}{2}\right)^{-1}\left(\mathrm{Id}-\frac{u}{2}\right)\left(\mathrm{Id}+\frac{u}{2}\right) &= v\\ \operatorname{dcay}_u(v) &= \left(\mathrm{Id}-\frac{u}{2}\right)^{-1} v\left(\mathrm{Id}+\frac{u}{2}\right)^{-1}.\end{aligned}$$

Similarly, for the inverse dcay_u^{-1}, one can invert the operator dcay_u which acts on the matrix X,

$$\mathrm{dcay}_u^{-1}(X) = (\mathrm{dcay}_u)^{-1}(X) = \left(\mathrm{Id} - \frac{u}{2}\right) X \left(\mathrm{Id} + \frac{u}{2}\right).$$

This can be expanded into the right hand side of Eq (3.7) involving the adjoint operator ad. □

An important note concerning the Cayley transform, is that it is not necessarily an approximation to the exponential map using the Padé Approximation. We can replace $\alpha = 1/2$ with another non-zero coefficient, and the Cayley transform is still valid [19].

3.4 Canonical Coordinate of the Second Kind

The Canonical coordinate of the second kind was developed by Owren and Marthinsen in [31]. The idea is to replace one single mapping of $\Psi = \exp$ with the product of the matrix exponentials based on the basis of the Lie algebra $\mathfrak{g}$. This can lead to a surprising reduction in computational cost, since each matrix exponential of a basis element can be relatively cheap to compute.

Suppose that $B = [e_1, \dots, e_n]$ is the basis of an n dimensional Lie algebra $\mathfrak{g}$. Then $u \in \mathfrak{g}$ is a linear combination of this basis,

$$u = u_1 e_1 + u_2 e_2 + \cdots + u_n e_n,$$

where $u_1, \dots, u_n$ are the coefficients. In this case, we can consider,

$$\exp(u_1 e_1 + \cdots + u_n e_n) = \exp(u_1 e_1) \exp(u_2 e_2) \cdots \exp(u_n e_n) \quad \in G.$$

The above is the definition for the **canonical coordinates of the second kind** with respect to the Lie algebra basis B, where the map $\mathrm{ccsk} : \mathfrak{g} \to G$ has the form,

$$\mathrm{ccsk}\left(\sum_{i=1}^{n} u_i e_i\right) = \exp(u_1 e_1) \exp(u_2 e_2) \cdots \exp(u_n e_n).$$

To derive the expression for the differential of ccsk, we use the right trivialisation, and consider the coefficients $[u_1, \dots, u_n]$, $[v_1, \dots, v_n]$ with respect to the basis B for $u, v \in \mathfrak{g}$. Suppose that we want to add the two elements $u, v \in \mathfrak{g}$ together, $u + tv$ for $t \in \mathbb{R}$. Let $(u_1 + tv_1)$ denotes the first coefficient with respect to the basis $e_1 \in B$. Then, the derivative of the element $\exp((u_1 + tv_1)e_1)$ is given by,

$$\frac{d}{dt} \exp((u_1 + tv_1)e_1)\bigg|_{t=0} = v_1 e_1 \exp((u_1 + tv_1)e_1)\bigg|_{t=0} = v_1 e_1 \exp(u_1 e_1).$$

Similarly, if we now consider all coefficients for the basis B, then the ccsk mapping for $u + tv$ is given by a product of the matrix exponentials with respect to B. This derivative of ccsk is,

$$\begin{aligned}
\frac{d}{dt}\mathrm{ccsk}\left(\sum_{i=1}^{n}(u_i + tv_i)e_i\right)\bigg|_{t=0} &= \frac{d}{dt}\left(\exp((u_1 + tv_1)e_1) \cdots \exp((u_n + tv_n)e_n)\right)\bigg|_{t=0} \\
&= v_1 e_1 \exp(u_1 e_1) \cdots \exp(u_n e_n) + \exp(u_1 e_1) v_2 e_2 \exp(u_2 e_2) \cdots \exp(u_n e_n) \\
&\quad + \cdots + \exp(u_1 e_1) \cdots \exp(u_{n-1} e_{n-1}) v_n e_n \exp(u_n e_n) \\
&= \left[v_1 e_1 + \sum_{i=1}^{n} v_i \mathrm{Ad}_{\exp(u_1 e_1)} \cdots \mathrm{Ad}_{\exp(u_{i-1} e_{i-1})}(e_i)\right] \cdot \exp\left(\sum_{i=1}^{n} u_i e_i\right)
\end{aligned}$$

using Ad in Definition 1.13.

So, the differential of the ccsk mapping is given by,

$$\mathrm{dccsk}_{\sum\limits_{i=1}^{n} u_i e_i}\left(\sum_{i=1}^{n} v_i e_i\right) = v_1 e_1 + \sum_{i=1}^{n} v_i \mathrm{Ad}_{\exp(u_1 e_1)} \cdots \mathrm{Ad}_{\exp(u_{i-1} e_{i-1})}(e_i).$$

However, solving for the inverse mapping dccsk^{-1} is a specialist subject in its own right, and will not be considered in this thesis. For further details, please refer to [31] and [19].

3.5 RKMK Order Conditions

The derivation of order conditions for the classical Runge-Kutta methods is based on the Taylor series expansion. This is done by matching the coefficients in the Taylor expansion of the approximate solution from the Runge-Kutta method, with the coefficients in the Taylor expansion of the exact solution [7].

On manifolds however, we need to use the Lie series introduced in Section 1.2.6 to replace the traditional Taylor series. This leads to the definition of a special form of the Lie series called the Lie-Butcher series, which was introduced by Munthe-Kaas in [26]. The Lie-Butcher series allows us to find the order conditions for the RKMK method in a similar way to the classical order conditions.

In [26], Munthe-Kaas showed that it is possible to derive the classical order conditions based on the coordinate-free setting, with respect to the manifold $\mathcal{M}$ and for abelian Lie algebra $\mathfrak{g}$. In the second paper [27], the Lie-Butcher order theory is developed further so that the order conditions can be derived to an arbitrary order for general Lie groups. In the rest of this chapter, we follow this approach to derive the order conditions, based on the RKMK Algorithm defined in Section 3.3.

Once the order conditions on the manifolds are found, it turns out that they are the same as the classical order conditions from the RK method, described in [6]. Hence this is another way to prove that Theorem 3.10 in Section 3.3 is correct.

However, before we can discuss the details in the construction of the order conditions for the RKMK method, we need to first mention the Universal enveloping algebra of $\mathfrak{g}$, which describes the higher order derivation operators, and the s-fold Cartesian product notations for the Lie group and Lie algebra.

3.5.1 Universal Enveloping Algebra $\mathfrak{G}$ of a Lie algebra $\mathfrak{g}$

The enveloping algebra $\mathfrak{G}$ allows for the introduction of higher order differential operators, which are needed in the derivations of the order conditions for the RKMK method [27].

Using the 'right' convention, recall that an element $v \in \mathfrak{g}$ defines the right invariant vector field at $a \in G$ as denoted in Section 1.3.2, then,

$$X_v(a) = R'_a(v).$$

In most cases, we can take $R'_a(v) = v \cdot a$, and the flow to this right invariant vector field is given by the 'left' multiplication similar to the general solution in Section 1.3.6,

$$\Psi_{X_v,t}(a) = L_{\exp(tv)}(a) = \exp(tv) \cdot a.$$

Let us consider f as a smooth real-valued function on the Lie group G for now. Then, an element $v \in \mathfrak{g}$ can act as a derivation of f from the right, as the Lie derivative of f with respect to the right invariant vector field X_v [27],

$$([f]v)(a) = \frac{\partial}{\partial t} f(\exp(tv) \cdot a)\bigg|_{t=0}. \tag{3.10}$$

We can then introduce higher order right invariant differential operators by iterating the first order operator in Eq (3.10). That is, we can introduce the second order right invariant operator as,

$$([f](u \cdot v))(a) = ([[f]u]v)(a) = \frac{\partial^2}{\partial s \partial t} f(\exp(su) \cdot \exp(tv) \cdot a)\bigg|_{t=0}. \tag{3.11}$$

Note that the expression $u \cdot v$ in the second order right invariant operator should not be confused with a matrix product. Also, the sum of two differential operators is defined as,

$$[f](u + v) = [f]u + [f]v,$$

and the zeroth order identity operator $\mathbb{I}$ is defined such that,

$$[f]\mathbb{I} = f.$$

All of the above leads to the following definition in [27].

Definition 3.15. *The Universal enveloping algebra $\mathfrak{G}$ of a Lie algebra $\mathfrak{g}$ consists of all invariant differential operators generated by $\{\mathbb{I}, \mathfrak{g}\}$ under the operations $\cdot$ and $+$ defined above.*

3.5.2 s-fold Cartesian Products of Lie Groups

Let G_s denote the s-fold Cartesian product of a Lie group, such that $G_s = G \times \cdots \times G$. Let $\mathfrak{g}_s$ be its Lie algebra $\mathfrak{g}_s = \mathfrak{g} \times \cdots \times \mathfrak{g}$, and let $\mathfrak{G}_s$ be the enveloping algebra, $\mathfrak{G}_s = \mathfrak{G} \times \cdots \times \mathfrak{G}$. The elements of G_s, $\mathfrak{g}_s$, and $\mathfrak{G}_s$ are s-tuples of G, $\mathfrak{g}$, and $\mathfrak{G}$ respectively, and all the basic operations are defined component-wise on the tuples. For example, if $v = (v_1, v_2, \ldots, v_s) \in \mathfrak{g}_s$, and each $v_i \in \mathfrak{g}$ for $i = 1, \ldots, s$, and likewise for $u \in \mathfrak{g}_s$, then we have [27],

$$\begin{aligned} \exp(v) &= (\exp(v_1), \ldots, \exp(v_s)), \\ u + v &= (u_1 + v_1, \ldots, u_s + v_s). \end{aligned}$$

Also, the tensor product basis for $\mathfrak{g}_s$ is introduced as [27],

$$\mathfrak{g}_s = \mathbb{R}^s \otimes \mathfrak{g},$$

such that $\mathfrak{g}_s$ is the linear span of vectors of the form,

$$q \otimes v = (q_1 v, q_2 v, \ldots, q_s v), \qquad \forall q \in \mathbb{R}^s, , v \in \mathfrak{g}.$$

Similarly, from Section 3.5.1, the identity operator in $\mathfrak{G}$ is denoted $\mathbb{I}$. If we have $\mathbf{1} = (1, 1, \ldots, 1)^T$, then the identity in the s-fold $\mathfrak{G}_s$ is given by,

$$\mathbb{I}_s = \mathbf{1} \otimes \mathbb{I} = (\mathbb{I}, \mathbb{I}, \ldots, \mathbb{I}).$$

Furthermore, the product in $\mathfrak{G}_s$ has the notation,

$$(q \otimes v) \cdot (r \otimes w) = (q \cdot r) \otimes (v \cdot w), \qquad \forall q, r \in \mathbb{R}^s, , v, w \in \mathfrak{G}. \tag{3.12}$$

Note that $(v \cdot w)$ is not the matrix product, but the notation for the higher order right invariant differential operator. Also, given that $q, r \in \mathbb{R}^s$, then $q \cdot r = r \cdot q$, but the differential operators v and w are generally non-commutating, that is $v \cdot w \neq w \cdot v$.

3.5.3 Lie-Butcher Series for the Analytic Solution

The Lie series, introduced at the end of Section 1.2.6, is used to provide a series expansion on manifolds. We will now look at the special form of the Lie series, called the Lie-Butcher series. This will be used to derive the series expansion of the analytic solution $v(t) \in \mathfrak{g}$ for $t \in \mathbb{R}$, which characterises the general initial value problem with respect to the right invariant vector field,

$$y' = R'_{y(t)}(f(y)), \qquad y(0) = p \in \mathcal{M},\ t \in \mathbb{R}, \tag{3.13}$$

where $f : \mathcal{M} = G \to \mathfrak{g}$, similar to the function in the generic presentation of differential equations in Eq (1.39).

Note that for this type of function f, the definitions of pull-back and derivation are the same as the definitions for the real-valued function $f : \mathcal{M} \to \mathbb{R}$ in Section 1.2.6 and Section 3.5.1. This is because we can consider the function in the derivation and pull-back definitions by components, with respect to some basis in the real vector space $\mathfrak{g}$. Hence, we can apply such definitions discussed in Section 1.2.6 and Section 3.5.1 to the Lie series for $f : \mathcal{M} \to \mathfrak{g}$.

In the following sections, we will use the notations $v^{(i)}$ and $([f]v)^{(i)}$, which are explained below.

As $t \in \mathbb{R}$ varies, $v(t) \in \mathfrak{g}$ traces out a curve in $\mathfrak{g}$. Its ith derivative, at $t = 0$, is given by,

$$v^{(i)} = \frac{\partial^i}{\partial t^i} v(t)\bigg|_{t=0}.$$

Similarly, for the Lie derivative of f ($f : \mathcal{M} \to \mathbb{R}$ or $f : \mathcal{M} \to \mathfrak{g}$) with respect to $v \in \mathfrak{g}$, the ith derivative of Eq (3.10), in $\mathfrak{G}_s$ is given by,

$$([f]v)^{(i)} = \frac{\partial}{\partial t}([f]v).$$

Let F be a vector field, $\Psi_{F,t}$ be the flow of F, and f be a smooth function, either $f : \mathcal{M} \to \mathbb{R}$ or $f : \mathcal{M} \to \mathfrak{g}$. $\Psi^*_{F,t} f$ denotes the pull-back of the object f along the flow of F. We continue to use the 'right' convention from Lie type equations in Section 1.3.6, such that for an arbitrary curve $y(t) \in \mathcal{M} = G$ as $t \in \mathbb{R}$ varies, $F(y(t)) = X_{v(t)}(y(t)) = R'_{y(t)}(v(t))$ is the right invariant vector field with respect to the time dependent curve $v(t) \in \mathfrak{g}$ for $t \in \mathbb{R}$.

To find the expression for the higher order derivation of f in the enveloping algebra $\mathfrak{G}$, we use Eq (1.11) and Eq (1.13) in Section 1.2.6, but with $v(t) \in \mathfrak{g}$ for $t \in \mathbb{R}$ acting as a derivation operator from the right as in Section 3.5.1. If f is independent of time t, then by Eq (1.11) we have the first order right invariant differential operator, for $t \in \mathbb{R}$ and $X_{v(t)}$ the right invariant vector field defined at the beginning of Section 3.5.1,

$$\frac{\partial}{\partial t}\left(\Psi^*_{X_{v(t)},t} f\right) = \Psi^*_{X_{v(t)},t}\left([f]X_{v(t)}\right) = \Psi^*_{X_{v(t)},t}\left([f]v(t)^{(0)}\right), \tag{3.14}$$

since for any $y \in \mathcal{M}$, the right invariant vector field is defined as $X_{v(t)}(y) = R'_y(v(t)) = v(t) \cdot y$, then we can substitute $X_{v(t)}$ with $v(t)$ in Eq (3.14). To find the second order right invariant differential operator, we differentiate Eq (3.14) according to Eq (1.13),

$$\begin{aligned}
\frac{\partial^2}{\partial t^2}\Big(\Psi^*_{X_{v(t)},t}f\Big) &= \frac{\partial}{\partial t}\Big(\Psi^*_{X_{v(t)},t}[f]v(t)^{(0)}\Big) \quad \text{substitute Eq (3.14)} \\
&= \Psi^*_{X_{v(t)},t}\Big([f]v(t)^{(0)}\cdot v(t)^{(0)} + \frac{\partial}{\partial t}[f]v(t)^{(0)}\Big) \\
&= \Psi^*_{X_{v(t)},t}\Big([f]v(t)^{(0)}\cdot v(t)^{(0)} + [f]v(t)^{(1)}\Big). \qquad (3.15)
\end{aligned}$$

Similarly, the third order right invariant differential operator can be found by substituting Eq (3.15) and using Eq (1.13),

$$\begin{aligned}
\frac{\partial^3}{\partial t^3}\Big(\Psi^*_{X_{v(t)},t}f\Big) &= \frac{\partial}{\partial t}\Big(\Psi^*_{X_{v(t)},t}\big([f]v(t)^{(0)}\cdot v(t)^{(0)} + [f]v(t)^{(1)}\big)\Big) \\
&= \Psi^*_{X_{v(t)},t}\Big([f]v(t)^{(0)}\cdot v(t)^{(0)}\cdot v(t)^{(0)} + [f]v(t)^{(1)}\cdot v(t)^{(0)} \\
&\quad + \frac{\partial}{\partial t}\big([f]v(t)^{(0)}\cdot v(t)^{(0)} + [f]v(t)^{(1)}\big)\Big) \\
&= \Psi^*_{X_{v(t)},t}\Big([f]v(t)^{(0)}\cdot v(t)^{(0)}\cdot v(t)^{(0)} + [f]2v(t)^{(1)}\cdot v(t)^{(0)} \\
&\quad + [f]v(t)^{(0)}\cdot v(t)^{(1)} + [f]v(t)^{(2)}\Big), \qquad (3.16)
\end{aligned}$$

and so on for other higher order differential operators.

Now, if we introduce the notation B^n, we have the following definition given below.

Definition 3.16. *For the nth right invariant differential operator on f which is time-independent,*

$$\frac{\partial^n}{\partial t^n}\Big(\Psi^*_{X_{v(t)},t}f\Big) = \Psi^*_{X_{v(t)},t}\Big([f]B^n(v(t))\Big), \qquad (3.17)$$

$B^n(v(t)) \in \mathfrak{G}$ is defined recursively as,

$$\begin{aligned}
B^0() &= \mathbb{I} \\
B^1(v(t)^{(0)}) &= v(t)^{(0)} \\
B^n(v(t)) &= B^{n-1}(v(t))\cdot v(t)^{(0)} + \frac{\partial}{\partial t}\Big(B^{n-1}(v(t))\Big). \qquad (3.18)
\end{aligned}$$

This means that for $n = 0, \ldots, 3$, we have the following expressions for B^n,

$$\begin{aligned}
B^0() &= \mathbb{I} \qquad (3.19) \\
B^1(v(t)^{(0)}) &= v(t)^{(0)} \qquad (3.20) \\
B^2(v(t)^{(0)}, v(t)^{(1)}) &= v(t)^{(0)}\cdot v(t)^{(0)} + v(t)^{(1)} \qquad (3.21) \\
B^3(v(t)^{(0)}, v(t)^{(1)}, v(t)^{(2)}) &= v(t)^{(0)}\cdot v(t)^{(0)}\cdot v(t)^{(0)} + 2v(t)^{(1)}\cdot v(t)^{(0)} \\
&\quad + v(t)^{(0)}\cdot v(t)^{(1)} + v(t)^{(2)} \qquad (3.22)
\end{aligned}$$

Then, the above right invariant differential operators in Eq (3.14) to Eq (3.16), can be rewritten as,

$$\begin{aligned}
\Psi^*_{X_{v(t)},t} f &= \Psi^*_{X_{v(t)},t}[f]\mathbb{I} \\
\frac{\partial}{\partial t}\left(\Psi^*_{X_{v(t)},t} f\right) &= \Psi^*_{X_{v(t)},t}\left([f]B^1(v(t)^{(0)})\right) \\
\frac{\partial^2}{\partial t^2}\left(\Psi^*_{X_{v(t)},t} f\right) &= \Psi^*_{X_{v(t)},t}\left([f]B^2(v(t)^{(0)}, v(t)^{(1)})\right) \\
\frac{\partial^3}{\partial t^3}\left(\Psi^*_{X_{v(t)},t} f\right) &= \Psi^*_{X_{v(t)},t}\left([f]B^3(v(t)^{(0)}, v(t)^{(1)}, v(t)^{(2)})\right).
\end{aligned}$$

We will now use Lemma 1.5 in Section 1.3.6 for Lie type equations, and the nth derivative $B^n(v(t))$ notation to modify the Lie series at the end of Section 1.2.6. This will lead to a series expansion of the analytical solution $v(t) \in \mathfrak{g}$ for $t \in \mathbb{R}$, which characterises the initial value problem given in Eq (3.13).

From Lemma 1.5 with respect to the Lie type Equation in Eq (1.40), we substitute in the general form of the initial value problem given by Eq (3.13), and find that $v(t) \in \mathfrak{g}$ is given by,

$$\begin{aligned}
v(t) &= R'_{y(t)^{-1}}(y'(t)) \\
&= R'_{y(t)^{-1}}(R'_{y(t)}(f(y(t)))) = f(y(t)) && \text{substitute Eq (3.13)} \\
&= f(s(t)\cdot p) && \text{substitute the fundamental solution} \\
&= f(L_{s(t)}(p)) = f(\Psi_{X_{v(t)},t}(p)) && \text{by Eq (1.42)} \\
&= \left(\Psi^*_{X_{v(t)},t} f\right)(p). && (3.23)
\end{aligned}$$

From Eq (3.17) and Eq (3.23), we have the following derivatives for $v(t) \in \mathfrak{g}$ at $t = 0$,

$$\begin{aligned}
v(0)^{(0)} &= \Psi^*_{X_{v(t)},t}\big([f]B^0()\big)\big|_{t=0}(p) = [f]B^0()(p) \\
v(0)^{(1)} &= \Psi^*_{X_{v(t)},t}\big([f]B^1(v(t)^{(0)})\big)\big|_{t=0}(p) = [f]B^1(v(0)^{(0)})(p) \\
v(0)^{(2)} &= \Psi^*_{X_{v(t)},t}\big([f]B^2(v(t)^{(0)}, v(t)^{(1)})\big)\big|_{t=0}(p) = [f]B^2(v(0)^{(0)}, v(0)^{(1)})(p) \\
&\vdots \\
v(0)^{(n)} &= \Psi^*_{X_{v(t)},t}\big([f]B^n(v(t)^{(0)}, \ldots, v(t)^{(n-1)})\big)\big|_{t=0}(p) \\
&= [f]B^n(v(0)^{(0)}, \ldots, v(0)^{(n-1)})(p),
\end{aligned}$$

where the flow Ψ at $t = 0$ is the identity map.

By the definition of the Lie series in Eq (1.16), and the above, we have a series expansion for $v(t)$ at $t = 0$ in terms of the higher order right invariant differential operators given above,

$$\begin{aligned}
\Psi^*_{X_{v(t)},t} f(p) &= \Psi^*_{X_{v(t)},t} f\Big|_{t=0}(p) + t\frac{\partial}{\partial t}\Psi^*_{X_{v(t)},t} f\Big|_{t=0}(p) + \frac{t^2}{2!}\frac{\partial^2}{\partial t^2}\Psi^*_{X_{v(t)},t} f\Big|_{t=0}(p) + \cdots \\
\Rightarrow v(t) &= v(0)^{(0)} + tv(0)^{(1)} + \frac{t^2}{2!}v(0)^{(2)} + \cdots
\end{aligned}$$

This is presented more formally in the theorem below from [27].

Theorem 3.11. *Let $y(t)$ be the solution to the initial value problem in Eq (3.13), and $v(t)$ be the corresponding curve on $\mathfrak{g}$ defined in Lemma 1.5. Then, the Lie-Butcher series characterising this IVP is given by,*

$$v(t) = \sum_{k=0}^{\infty} \frac{t^k}{k!} v(0)^{(k)},$$

where $v(0)^{(k)} = [f]B^{(k)}(v(0)^{(0)}, v(0)^{(1)}, \dots, v(0)^{(k-1)})(p)$, and $B^k()$ is given by Eq (3.18).

To solidify the connection between the Lie-Butcher series and the classical Taylor expansion, Munthe-Kaas introduced the 'bar' operator for the higher order differential operators in the Lie-Butcher series. Let the 'bar' operator denote,

$$\overline{v(0)} = [f]v(0)^{(0)}(p),$$

such that the bar hides the notations $[f]$ and (p) within its definition. Note that this maps the differential operator in $\mathfrak{G}$ back to $\mathfrak{g}$. That is, the higher order differential operators can be rewritten in the shorthand, according to Eq (3.19) to Eq (3.22) and Theorem 3.11,

$$
\begin{aligned}
v(0)^{(0)} &= [f]B^0()(p) = [f]\mathbb{I}(p) = \overline{\mathbb{I}} && (3.24)\\
v(0)^{(1)} &= [f]B^1(v(0)^{(0)})(p) = [f]v(0)^{(0)}(p) = [f]\overline{\mathbb{I}}(p) = \overline{\overline{\mathbb{I}}} && (3.25)\\
v(0)^{(2)} &= [f]B^2(v(0)^{(0)}, v(0)^{(1)})(p) = [f](v(0)^{(0)} \cdot v(0)^{(0)} + v(0)^{(1)})(p) &&\\
&= [f](\overline{\mathbb{I}} \cdot \overline{\mathbb{I}} + \overline{\overline{\mathbb{I}}})(p) = \overline{\overline{\mathbb{I}} \cdot \overline{\mathbb{I}}} + \overline{\overline{\overline{\mathbb{I}}}} && (3.26)\\
v(0)^{(3)} &= [f]B^3(v(0)^{(0)}, v(0)^{(1)}, v(0)^{(2)})(p) &&\\
&= [f](v(0)^{(0)} \cdot v(0)^{(0)} \cdot v(0)^{(0)} + 2v(0)^{(1)} \cdot v(0)^{(0)} + v(0)^{(0)} \cdot v(0)^{(1)} + v(0)^{(2)})(p) &&\\
&= [f](\overline{\mathbb{I}} \cdot \overline{\mathbb{I}} \cdot \overline{\mathbb{I}} + 2\overline{\overline{\mathbb{I}}} \cdot \overline{\mathbb{I}} + \overline{\mathbb{I}} \cdot \overline{\overline{\mathbb{I}}} + \overline{\overline{\mathbb{I}} \cdot \overline{\mathbb{I}}} + \overline{\overline{\overline{\mathbb{I}}}})(p) &&\\
&= \overline{\overline{\mathbb{I}} \cdot \overline{\mathbb{I}} \cdot \overline{\mathbb{I}}} + 2\overline{\overline{\overline{\mathbb{I}}} \cdot \overline{\mathbb{I}}} + \overline{\overline{\mathbb{I}} \cdot \overline{\overline{\mathbb{I}}}} + \overline{\overline{\overline{\mathbb{I}} \cdot \overline{\mathbb{I}}}} + \overline{\overline{\overline{\overline{\mathbb{I}}}}} && (3.27)
\end{aligned}
$$

Furthermore, if we invert the bar pattern upside down, this yields the relationship between these operators and the elementary differentials in the traditional Butcher theory, represented by rooted trees [27]. Starting with a differential equation,

$$y'(t) = \mathbf{f}(y(t)),$$

The elementary differentials are the compositions of the derivatives of $\mathbf{f}$, which are needed for the Taylor series expansions of solutions when deriving the classical order conditions [7].

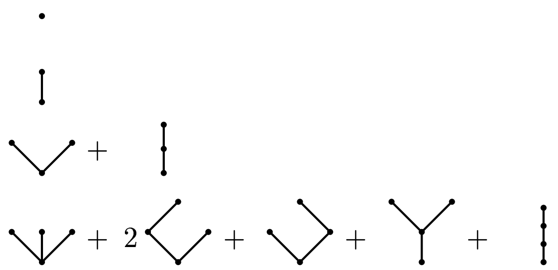
So, with the rooted trees to represent the higher order differential operators, Eq (3.24) to Eq (3.25) can be rewritten as,

$v(0)^{(0)} =$

$v(0)^{(1)} =$

$v(0)^{(2)} =$ … $+$ …

$v(0)^{(3)} =$ … $+ 2$ … $+$ … $+$ … $+$ …

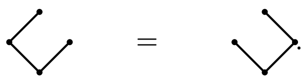


which gives the first part of the order conditions of up to order 4, for the RKMK method.

Table 3.1 displays the trees for both the traditional Runge-Kutta method and for the Runge-Kutta-Munthe-Kaas method up to order 4. Note that in the classical Butcher theory, the manifold $\mathcal{M}$ and the Lie group G are defined to be the real vector space $\mathbb{R}^n$ [26], and the group product is the sum of vectors [29]. The Lie algebra $\mathfrak{g}$ is spanned by the infinitesimal generators $v_{\mathcal{M}} = \frac{\partial}{\partial x^i}$, so that $\mathfrak{g}$ is abelian, $\left[\frac{\partial}{\partial x^i}, \frac{\partial}{\partial x^j}\right] = 0$. Due to this commutativity in $\mathfrak{g}$, trees with permuted branches are consider identical. For example, for the Runge-Kutta method of order 4, we know that,

$$\mathbf{f}''(\mathbf{f}'\mathbf{f}, \mathbf{f}) = \mathbf{f}''(\mathbf{f}, \mathbf{f}'\mathbf{f}).$$

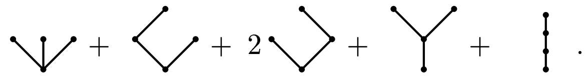
Hence, the two rooted trees below are considered identical.



However, when the group under consideration is non-commutative, as in most cases when using the RKMK method, then the permutation of the branches in the trees are considered separately. In other words, we can no longer treat all branch permutations as the same tree. Also, if we based the above analysis on the 'left' invariant vector field as in the original paper by Munthe-Kaas [27], then in some cases, we will have differential coefficients for some of the permutation of the branches in the rooted trees. For example, using the left invariant vector field, the third order differential operator $v(t)$ at $t = 0$ is given as,

$$v(0)^{(3)} = \overline{\overline{\mathbb{I}}\cdot\overline{\mathbb{I}}\cdot\overline{\mathbb{I}}} + \overline{\overline{\overline{\mathbb{I}}}\cdot\overline{\mathbb{I}}} + 2\overline{\overline{\mathbb{I}}\cdot\overline{\overline{\mathbb{I}}}} + \overline{\overline{\overline{\mathbb{I}}\cdot\overline{\mathbb{I}}}} + \overline{\overline{\overline{\overline{\mathbb{I}}}}}$$

$=$

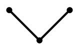
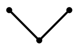
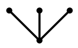
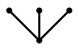
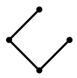
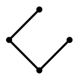
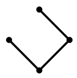
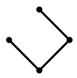
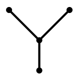
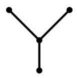


3.5.4 Lie-Butcher Series for the Approximate Solution

In the previous section, we gave a brief discussion on how to construct the Lie-Butcher series characterising the initial value problem in Eq (3.13). This gives the analytic solution $v(t) \in \mathfrak{g}$ for $t \in \mathbb{R}$, and leads to the relationship between the bar identity operators $\overline{\mathbb{I}} \in \mathfrak{G}$ and the elementary differentials in the classical Butcher tree theory displayed in Table 3.1. Now we want to construct the Lie-Butcher series for the approximate solution to v, based on the structure of the RKMK algorithm in Section 3.3, which will be dependent on the stepsize h. This approximate solution in $\mathfrak{g}$ is denoted $\widehat{v}(h)$.

The classical way of defining the order of a Runge-Kutta method on $\mathbb{R}^n$, is to say that the method has order q if $\|y - \widehat{y}\| = O(h^{q+1})$, where y is the analytic solution and $\widehat{y}$ is the numerical approximation dependent on stepsize h. Similarly, in the RKMK method, if we consider a curve $\widehat{y}(h) \in \mathcal{M}$, then there exists a corresponding curve $\widehat{v}(h) \in \mathfrak{g}$ via Lemma 1.5. So, the definition of order for the RKMK method can be found by equating the Lie-Butcher series for the analytic curve $v(t) \in \mathfrak{g}$, with the series from the numerical approximation $\widehat{v}(h) \in \mathfrak{g}$ from the RKMK algorithm. Formally, Munthe-Kaas presented this in the following definition [27].

Table 3.1: Rooted trees for RK and RKMK methods.

RK rooted trees			RKMK rooted trees		
order	tree	elementary differentials	order	tree	bar identity operator in $\mathfrak{G}$
1		$\mathbf{f}$	1		$\overline{\mathbb{I}}$
2		$\mathbf{f}'\mathbf{f}$	2		$\overline{\overline{\mathbb{I}}}$
3		$\mathbf{f}''(\mathbf{f},\mathbf{f})$	3		$\overline{\overline{\mathbb{I}}\cdot\overline{\mathbb{I}}}$
3		$\mathbf{f}'\mathbf{f}'\mathbf{f}$	3		$\overline{\overline{\overline{\mathbb{I}}}}$
4		$\mathbf{f}'''(\mathbf{f},\mathbf{f},\mathbf{f})$	4		$\overline{\overline{\mathbb{I}}\cdot\overline{\mathbb{I}}\cdot\overline{\mathbb{I}}}$
4		$\left.\right\}\ \mathbf{f}''(\mathbf{f}'\mathbf{f},\mathbf{f})$	4		$\overline{\overline{\overline{\mathbb{I}}}\cdot\overline{\mathbb{I}}}$
4			4		$\overline{\overline{\mathbb{I}}\cdot\overline{\overline{\mathbb{I}}}}$
4		$\mathbf{f}'\mathbf{f}''(\mathbf{f},\mathbf{f})$	4		$\overline{\overline{\overline{\mathbb{I}}\cdot\overline{\mathbb{I}}}}$
4		$\mathbf{f}'\mathbf{f}'\mathbf{f}'\mathbf{f}$	4		$\overline{\overline{\overline{\overline{\mathbb{I}}}}}$

Definition 3.17. *Let $y(t)$ be the analytic solution to the IVP in Eq (3.13), and let $\widehat{y}(h)$ be the numerical solution after one step of the RKMK algorithm, considered as a function of the stepsize h. Also, $v(t)$ and $\widehat{v}(h)$ are the corresponding curves in $\mathfrak{g}$ given by Lemma 1.5 :*

$$\begin{aligned} v(t) &= R'_{y(t)^{-1}}(y'(t)), \\ \widehat{v}(h) &= R'_{\widehat{y}(h)^{-1}}(\widehat{y}'(h)). \end{aligned}$$

Then, the method has order q if,

$$v^{(i)} = \widehat{v}^{(i)}, \quad \textit{for } i = 0, \ldots, q-1.$$

Also, it is convenient to analyse the order of the RKMK method if the RKMK algorithm is written using the s-fold cartesian product notation. Let us now introduce the following s-fold products : $G_s = G \times \cdots \times G$, $f_s = f \times \cdots \times f$, and the basic operations act in a component-wise fashion. Let $\mathfrak{g}_s$ be the corresponding Lie algebra of G_s, such that $\mathfrak{g}_s = \mathbb{R}^s \otimes \mathfrak{g}$, and $\mathfrak{G}_s$ is the enveloping algebra of $\mathfrak{g}_s$.

Hence, we have the notations for the RK coefficients in the RKMK algorithm, given by the following,

$$A_s = A \otimes I, \qquad b_s = b^T \otimes I, \qquad p_s = (p, \ldots, p) = (y(0), \ldots, y(0)),$$

and we can rewrite the s-stage RKMK algorithm into a 1-stage method by 'stacking' the internal stages together. This means that we now have,

$$\begin{aligned}
u &= \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_s \end{bmatrix} = h \begin{bmatrix} a_{11}\mathrm{Id} & a_{12}\mathrm{Id} & \cdots & a_{1s}\mathrm{Id} \\ a_{21}\mathrm{Id} & \ddots & & \vdots \\ \vdots & & \ddots & \\ a_{s1}\mathrm{Id} & \cdots & \cdots & a_{ss}\mathrm{Id} \end{bmatrix} \begin{bmatrix} \widetilde{k}_1 \\ \widetilde{k}_2 \\ \vdots \\ \widetilde{k}_s \end{bmatrix} = h(A \otimes \mathrm{Id})\widetilde{k} = hA_s\widetilde{k} \\
k &= \begin{bmatrix} k_1 \\ k_2 \\ \vdots \\ k_s \end{bmatrix} = \begin{bmatrix} f(\Lambda(\exp(u_1), p)) \\ f(\Lambda(\exp(u_2), p)) \\ \vdots \\ f(\Lambda(\exp(u_s), p)) \end{bmatrix} = f_s(\Lambda(\exp(u), p_s)) = f_s(\exp(u) \cdot p_s) \\
&= f_s(L_{\exp(u)}(p_s)) = (L^*_{\exp(u)} f_s)(p_s) \\
\widetilde{k} &= \begin{bmatrix} \widetilde{k}_1 \\ \widetilde{k}_2 \\ \vdots \\ \widetilde{k}_s \end{bmatrix} = \begin{bmatrix} \mathrm{dexp}_{u_1}(k_1) \\ \mathrm{dexp}_{u_2}(k_2) \\ \vdots \\ \mathrm{dexp}_{u_s}(k_s) \end{bmatrix} = \mathrm{dexp}_u(k) \\
v &= h(b_1\widetilde{k}_1 + b_2\widetilde{k}_2 + \cdots + b_s\widetilde{k}_s) = h(b_1\mathrm{Id}, b_2\mathrm{Id}, \ldots, b_s\mathrm{Id})\widetilde{k} = h(b^T \otimes \mathrm{Id})\widetilde{k} = hb_s\widetilde{k} \\
\widehat{y}_1 &= \Lambda(\exp(v), p) = \exp(v) \cdot p = L_{\exp(v)}(p).
\end{aligned}$$

So, the s-fold cartesian product of the 1 stage RKMK method is,

$$\begin{aligned}
p_s &= (p, \ldots, p) \\
u &= hA_s\widetilde{k} \qquad u \in \mathfrak{g}_s \\
k &= f_s(L_{\exp(u)}(p_s)) = (L^*_{\exp(u)} f_s)(p_s) \qquad v \in \mathfrak{g}_s \\
\widetilde{k} &= \mathrm{dexp}_u(k) \qquad \widetilde{k} \in \mathfrak{g}_s \\
v &= hb_s\widetilde{k} \qquad v \in \mathfrak{g} \\
\widehat{y}_1 &= L_{\exp(v)}(p) \qquad \widehat{y}_1 \in G.
\end{aligned} \tag{3.28}$$

Recall that we use $\widehat{v}(h)$ to denote the numerical solution in $\mathfrak{g}$ which characterises the IVP in Eq (3.13). Then similar to the Lie-Butcher series for the analytic solution $v(t)$ in Theorem 3.11, the Lie-Butcher series for $\widehat{v}(h)$ is given by,

$$\widehat{v}(h) = \widehat{v}(0)^{(0)} + h\widehat{v}(0)^{(1)} + \frac{h^2}{2!}\widehat{v}(0)^{(2)} + \cdots \tag{3.29}$$

where $\widehat{v}(0)^{(i)}$ denotes the ith derivative of $\widehat{v}(h)$ with respect to stepsize h at $h = 0$. If we can find the expressions for $\widehat{v}^{(i)}$, then we can derive the order conditions for the RKMK method, by matching these derivatives with the derivatives $v^{(i)}$ in the analytic Lie-Butcher series, according to Definition 3.17. But before we can do this, we need to discuss the following Lemma from [27].

Lemma 3.11. *Let $\widehat{v}(h) \in \mathfrak{g}$ be defined by Definition 3.17, then*

$$\widehat{v}(h) = \mathrm{dexp}_v(v^{(1)}),$$

where $v = hb_s\widetilde{k}$ and $v^{(1)}$ is the derivative with respect to h.

Let $\widehat{z}(h) = \exp(u) \in G_s$, $u = hA_s\widetilde{k} \in \mathfrak{g}_s$, and $\widehat{u}(h)$ is the numerical solution similarly defined by Definition 3.17, such that we have, $\widehat{u}(h) = R'_{\widehat{z}(h)^{-1}}(\widehat{z}'(h))$. Then,

$$\widehat{u}(h) = \mathrm{dexp}_u(u^{(1)}).$$

Proof. To prove this for $\widehat{v}(h) \in \mathfrak{g}$, we start with Definition 3.17, where,

$$\widehat{v}(h) = R'_{\widehat{y}(h)^{-1}}(\widehat{y}'(h)),$$

From the s-fold RKMK Algorithm in Eq (3.28), we know that $\widehat{y}(h) = \exp(v) \cdot p \in G$. If we substitute this into the above, and use Theorem 1.3 for the derivative of the exponential map, we have,

$$\begin{aligned} \widehat{v}(h) &= R'_{(\exp(v)\cdot p)^{-1}}\left(\frac{\partial}{\partial h}\exp(v)\cdot p\right) = R'_{(\exp(v))^{-1}} \cdot R'_{p^{-1}}\left(R'_p \frac{\partial}{\partial h}\exp(v)\right) \\ &= R'_{\exp(-v)} \cdot \frac{\partial}{\partial h}\exp(v) = R'_{\exp(-v)} \cdot R'_{\exp(v)} \cdot \mathrm{dexp}_v\left(\frac{\partial v}{\partial h}\right) = \mathrm{dexp}_v(v^{(1)}). \end{aligned}$$

Similarly with the curve $\widehat{u}(h) \in \mathfrak{g}_s$, since $\widehat{z}(h) = \exp(u) \in G_s$, then we substitute this into Definition 3.17,

$$\begin{aligned} \widehat{u}(h) &= R'_{\widehat{z}(h)^{-1}}(\widehat{z}'(h)) = R'_{\exp(-u)}\left(\frac{\partial}{\partial h}\exp(u)\right) \\ &= R'_{\exp(-u)} \cdot R'_{\exp(u)} \cdot \mathrm{dexp}_u\left(\frac{\partial u}{\partial h}\right) = \mathrm{dexp}_u(u^{(1)}). \end{aligned}$$

□

Furthermore, from Lemma 3.11, given that $v = hb_s\widetilde{k}$ and $u = hA_s\widetilde{k}$ in the s-fold RKMK algorithm, we can make the observation that when $h = 0$, $v(0) = 0$ and $u(0) = 0$. So $\widehat{v}(0)$ and $\widehat{u}(0)$ in Lemma 3.11 are now given by,

$$\begin{aligned} \widehat{v}(0) &= \mathrm{dexp}_{v(0)}(v(0)^{(1)}) = v(0)^{(1)}, \\ \widehat{u}(0) &= \mathrm{dexp}_{u(0)}(u(0)^{(1)}) = u(0)^{(1)}. \end{aligned}$$

For the higher order derivatives of $\widehat{v}$ and $\widehat{u}$ at $h=0$, we have,

$$\widehat{v}(0)^{(i)} = v(0)^{(i+1)}, \tag{3.30}$$
$$\widehat{u}(0)^{(i)} = u(0)^{(i+1)}. \tag{3.31}$$

Now, to find the expressions for the derivatives of u and v at $h=0$ based on the s-fold RKMK algorithm, we need to use the **Leibniz rule**, which states that,

For a function $F(x)=f(x)g(x)$, the nth derivative of $F(x)$ with respect to x is given by,

$$F^{(n)}(x)=\sum_{k=0}^{n}\left(\begin{array}{c} n \\ k \end{array}\right) f^{(k)}(x)g^{(n-k)}(x)=\sum_{k=0}^{n}\frac{n!}{k!(n-k)!}f^{(k)}(x)g^{(n-k)}(x).$$

When applied to a function of the form, $F(h)=h\phi(h)$ at $h=0$, we have,

$$F^{(n)}(h)|_{h=0}=n\phi(h)^{(n-1)}|_{h=0}.$$

So, for $v=hb_s\widetilde{k}\in\mathfrak{g}$ and $u=hA_sk\in\mathfrak{g}_s$, their derivatives at $h=0$ are [27],

$$v(0)^{(i)} = i\frac{\partial^{(i-1)}}{\partial h^{(i-1)}}(b_s\widetilde{k})\bigg|_{h=0} = ib_s\widetilde{k}^{(i-1)}\bigg|_{h=0}, \tag{3.32}$$
$$u(0)^{(i)} = i\frac{\partial^{(i-1)}}{\partial h^{(i-1)}}(A_s\widetilde{k})\bigg|_{h=0} = iA_s\widetilde{k}^{(i-1)}\bigg|_{h=0}. \tag{3.33}$$

Combining Eq (3.30) and Eq (3.31) with Eq (3.32) and Eq (3.32) we have the numerical solutions $\widehat{v}$ and $\widehat{u}$ in terms of the coefficients in the s-fold RKMK algorithm,

$$\widehat{v}(0)^{(i)} = (i+1)b_s\widetilde{k}^{(i)}\big|_{h=0}, \tag{3.34}$$
$$\widehat{u}(0)^{(i)} = (i+1)A_s\widetilde{k}^{(i)}\big|_{h=0}. \tag{3.35}$$

From the two equations above, we now know that $\widehat{v}(0)$ and $\widehat{u}(0)$ are dependent on the derivative of $\widetilde{k}(h)$ with respect to h at $h=0$. Since $\widetilde{k}(h)=\mathrm{dexp}_{u(h)}(k(h))\in\mathfrak{g}_s$ in the s-fold RKMK algorithm, when $h=0$, $\widetilde{k}(0)=0$, and $\widetilde{k}(0)=k(0)$. By a similar argument as in Eq (3.30), we find,

$$\widetilde{k}(0)^{(i)}=k(0)^{(i)}, \qquad \text{for } h=0.$$

Hence, Eq (3.34) and (3.35) can both be rewritten as,

$$\widehat{v}(0)^{(i)} = (i+1)b_sk(0)^{(i)}, \tag{3.36}$$
$$\widehat{u}(0)^{(i)} = (i+1)A_sk(0)^{(i)}. \tag{3.37}$$

Finally, the expression for $k(0)^{(i)}$ is presented in the following theorem in [27].

Theorem 3.12. *The Lie-Butcher series in Eq (3.29) for the numerical solution $\widehat{v}(h) \in \mathfrak{g}$ corresponding to the solution $\widehat{y}(h) \in G$ is given by the recursion in $\widehat{v}^{(i)}$ in Eq (3.36), where $k(0)^{(i)}$ has the form,*

$$\begin{aligned} k(0)^{(i)} &= \overline{B^i(A_s k^{(0)}, 2A_s k^{(1)}, \ldots, iA_s k^{(i-1)})} \\ &= [f_s]B^i(A_s k^{(0)}, 2A_s k^{(1)}, \ldots, iA_s k^{(i-1)})(p_s). \end{aligned}$$

Note that the notation $B^i()$ was introduced in Section 3.5.3. In the s-fold cartesian structure, when $\widehat{u}(0)^{(i)} = (i+1)A_s k(0)^{(i)}$, then

$$\begin{aligned} B^0() &= \mathbb{I}_s \\ B^1(\widehat{u}(0)^{(0)}) &= \widehat{u}(0)^{(0)}, \end{aligned}$$

and the ith derivative is similar to Eq (3.18),

$$\begin{aligned} &B^i(\widehat{u}(0)^{(0)}, \widehat{u}(0)^{(1)}, \ldots, \widehat{u}(0)^{(i-1)}) = \\ &\quad B^{i-1}(\widehat{u}(0)^{(0)}, \widehat{u}(0)^{(1)}, \ldots, \widehat{u}^{(i-2)}) \cdot \widehat{u}(0)^{(0)} + \frac{\partial}{\partial h}\Big(B^{i-1}(\widehat{u}(0)^{(0)}, \widehat{u}(0)^{(1)}, \ldots, \widehat{u}^{(i-2)})\Big). \end{aligned}$$

Proof. To prove Theorem 3.12, we start from Lemma 3.11 and use Lemma 1.5 to find an expression for $k(0)^{(0)}$.

We define the curve $\widehat{z}(h) = \exp(u) \in G_s$ where $u = hA_s\widetilde{k}$, and in Lemma 3.11, we have,

$$\widehat{u}(h) = R'_{\widehat{z}(h)^{-1}}(\widehat{z}'(h)).$$

Then by Lemma 1.5, the flow which represents the general solution $\widehat{y}(h)$ to the IVP,

$$\widehat{y}'(h) = X_{\widehat{u}(h)}(\widehat{y}(h)) = R'_{\widehat{y}(h)}(\widehat{u}(h)), \qquad \widehat{y}(0) = p_s,$$

is given by,

$$\Psi_{X_{\widehat{u}(h)},h}(p_s) = L_{\widehat{z}(h)}(p_s).$$

Note that in this case, $\widehat{z}(h) = \exp(u)$ is the fundamental solution satisfying,

$$\widehat{z}'(h) = X_{\widehat{u}(h)}(\widehat{z}(h)), \qquad z(0) = e.$$

So, we can rewrite $k(h)^{(0)}$ as the pull-back of the flow with respect to this right invariant vector field $X_{\widehat{u}(h)}$,

$$k(h)^{(0)} = (L^*_{\exp(u)}f_s)(p_s) = (L^*_{\widehat{z}(h)}f_s)(p_s) = (\Psi^*_{X_{\widehat{u}(h)},h}f_s)(p_s).$$

To differentiate $k(h)$ with respect to h at $h = 0$, we can use Eq (3.17) so that,

$$\begin{aligned} \frac{\partial^n}{\partial h^n}\left(\Psi^*_{X_{\widehat{u}(h)},h}f_s\right)\Big|_{h=0} &= \Psi^*_{X_{\widehat{u}(h)},h}\left([f_s]B^n(\widehat{u}(h))\right)\Big|_{h=0} \\ &= [f_s]B^n(\widehat{u}(0)^{(0)}, \widehat{u}(0)^{(1)}, \ldots, \widehat{u}(0)^{(n-1)}). \end{aligned}$$

Hence, along with Eq (3.37), the nth derivative of $k(h)$ at $h = 0$ is given by,

$$\begin{aligned} k(0)^{(n)} &= \frac{\partial^n}{\partial h^n}\left(\Psi^*_{X_{\widehat{u}(h)},h}f_s\right)\Big|_{h=0}(p_s) \\ &= [f_s]B^n(\widehat{u}(0)^{(0)}, \widehat{u}(0)^{(1)}, \ldots, \widehat{u}(0)^{(n-1)})(p_s) \\ &= [f_s]B^n(A_s k(0)^{(0)}, 2A_s k(0)^{(1)}, \ldots, nA_s k(0)^{(n-1)})(p_s) \\ &= \overline{B^n(A_s k(0)^{(0)}, 2A_s k(0)^{(1)}, \ldots, nA_s k(0)^{(n-1)})}. \end{aligned}$$

$\square$

From Theorem 3.12, we now have the following expressions for the derivatives of $k(0)$ based on a 4th order Runge-Kutta method,

$$
\begin{aligned}
k(0)^{(0)} &= \overline{B^0()} = \bar{\mathbb{I}}_s \\
k(0)^{(1)} &= \overline{B^1(A_s k(0)^{(0)})} = \overline{A_s k(0)^{(0)}} \\
k(0)^{(2)} &= \overline{B^2(A_s k(0)^{(0)}, 2A_s k(0)^{(1)})} = \overline{A_s k(0)^{(0)} \cdot A_s k(0)^{(0)} + 2A_s k(0)^{(1)}} \\
k(0)^{(3)} &= \overline{B^3(A_s k(0)^{(0)}, 2A_s k(0)^{(1)}, 3A_s k(0)^{(2)})} \\
&= \overline{A_s k(0)^{(0)} \cdot A_s k(0)^{(0)} \cdot A_s k(0)^{(0)} + 2(2A_s k(0)^{(1)})A_s k(0)^{(0)}} \\
&\quad \overline{+A_s k(0)^{(0)} \cdot 2A_s k(0)^{(1)} + 3A_s k(0)^{(2)}}.
\end{aligned}
$$

Recall that in Section 3.5.2, we introduced the s-fold cartesian product notations. We can now include the bar notation as well, for $q \in \mathbb{R}^s$ and $v \in \mathfrak{G}$,

$$
\begin{aligned}
\overline{q \otimes v} &= [f_s](q \otimes v)(p_s) = [f, \ldots, f](q_1 v, \ldots, q_s v)(p, \ldots, p) \\
&= (q_1 \bar{v}, \ldots, q_s \bar{v}) = (q_1, \ldots, q_s) \otimes \bar{v} = q \otimes \bar{v}.
\end{aligned}
$$

Also, from the classical Butcher order conditions, $\sum a_{ij} = c_i$ can be rewritten as,

$$
A\mathbf{1} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1s} \\ a_{21} & \ddots & & \vdots \\ \vdots & & \ddots & \\ a_{s1} & \cdots & \cdots & a_{ss} \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^{s} a_{1j} \\ \sum_{j=1}^{s} a_{2j} \\ \vdots \\ \sum_{j=1}^{s} a_{sj} \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_s \end{bmatrix} = c.
$$

Using the above notation, the derivatives of $k(0)$ up to order 4 are now given in terms of the bar identity operators,

$$
\begin{aligned}
k(0)^{(0)} &= \bar{\mathbb{I}}_s = \overline{\mathbf{1} \otimes \mathbb{I}} = \mathbf{1} \otimes \bar{\mathbb{I}} && (3.38) \\
k(0)^{(1)} &= \overline{A_s k(0)^{(0)}} = \overline{(A \otimes I)(\mathbf{1} \otimes \bar{\mathbb{I}})} = \overline{A\mathbf{1} \otimes \bar{\mathbb{I}}} = c \otimes \bar{\bar{\mathbb{I}}} && (3.39) \\
k(0)^{(2)} &= \overline{A_s k(0)^{(0)} \cdot A_s k(0)^{(0)} + 2A_s k(0)^{(1)}} \\
&= \overline{c \cdot c \otimes \bar{\mathbb{I}} \cdot \bar{\mathbb{I}}} + \overline{2Ac \otimes \bar{\bar{\mathbb{I}}}} = c \cdot c \otimes \overline{\bar{\mathbb{I}} \cdot \bar{\mathbb{I}}} + 2Ac \otimes \bar{\bar{\bar{\mathbb{I}}}} && (3.40) \\
k(0)^{(3)} &= \overline{A_s k(0)^{(0)} \cdot A_s k(0)^{(0)} \cdot A_s k(0)^{(0)} + 2(2A_s k(0)^{(1)})A_s k(0)^{(0)}} \\
&\quad \overline{+A_s k(0)^{(0)} \cdot 2A_s k(0)^{(1)} + 3A_s k(0)^{(2)}} \\
&= c \cdot c \cdot c \otimes \overline{\bar{\mathbb{I}} \cdot \bar{\mathbb{I}} \cdot \bar{\mathbb{I}}} + 4((Ac) \cdot c) \otimes \overline{\bar{\bar{\mathbb{I}}} \cdot \bar{\mathbb{I}}} + 2(c \cdot (Ac)) \otimes \overline{\bar{\mathbb{I}} \cdot \bar{\bar{\mathbb{I}}}} \\
&\quad +3Ac \cdot c \otimes \overline{\overline{\bar{\mathbb{I}} \cdot \bar{\mathbb{I}}}} + 6A^2 c \otimes \overline{\bar{\mathbb{I}} \cdot \bar{\bar{\mathbb{I}}}} \\
&= c \cdot c \cdot c \otimes \overline{\bar{\mathbb{I}} \cdot \bar{\mathbb{I}} \cdot \bar{\mathbb{I}}} + 2c \cdot (Ac) \otimes \left[2\overline{\bar{\bar{\mathbb{I}}} \cdot \bar{\mathbb{I}}} + \overline{\bar{\mathbb{I}} \cdot \bar{\bar{\mathbb{I}}}} \right] \\
&\quad +3Ac \cdot c \otimes \overline{\overline{\bar{\mathbb{I}} \cdot \bar{\mathbb{I}}}} + 6A^2 c \otimes \overline{\bar{\mathbb{I}} \cdot \bar{\bar{\mathbb{I}}}} && (3.41)
\end{aligned}
$$

If we substitute Eq (3.38) to Eq (3.41) into the derivatives of $k(h)$ at $h = 0$ in Eq(3.36), then up to order 4, we have,

$$
\begin{aligned}
\widehat{v}(0)^{(0)} &= b_s k(0)^{(0)} = (b^T \otimes I)(\mathbf{1} \otimes \overline{\mathbb{I}}) = b^T \mathbf{1} \otimes \overline{\mathbb{I}} && (3.42)\\
\widehat{v}(0)^{(1)} &= 2b_s k(0)^{(1)} = 2(b^T \otimes I)(c \otimes \overline{\overline{\mathbb{I}}}) = 2b^T c \otimes \overline{\overline{\mathbb{I}}} && (3.43)\\
\widehat{v}(0)^{(2)} &= 3b_s k(0)^{(2)} = 3b^T c^2 \otimes \overline{\overline{\mathbb{I}} \cdot \overline{\mathbb{I}}} + 6b^T Ac \otimes \overline{\overline{\overline{\mathbb{I}}}} && (3.44)\\
\widehat{v}(0)^{(3)} &= 4b_s k(0)^{(3)}\\
&= 4b^T c^3 \otimes \overline{\overline{\mathbb{I}} \cdot \overline{\mathbb{I}} \cdot \overline{\mathbb{I}}} + 8b^T c \cdot Ac \otimes \left[2\overline{\overline{\overline{\mathbb{I}}} \cdot \overline{\mathbb{I}}} + \overline{\overline{\mathbb{I}} \cdot \overline{\overline{\mathbb{I}}}} \right]\\
&\quad +12b^T Ac^2 \otimes \overline{\overline{\overline{\overline{\mathbb{I}}} \cdot \overline{\mathbb{I}}}} + 24b^T A^2 c \otimes \overline{\overline{\overline{\overline{\mathbb{I}}}}} && (3.45)
\end{aligned}
$$

From Definition 3.17, for a method of order q, we require that $v(0)^{(i)} = \widehat{v}(0)^{(i)}$. So, for a 4th order method, if we compare the coefficients in Eq (3.24-3.27) to those in Eq (3.42-3.45), we can derive the order conditions for the RKMK method, listed in Table 3.2.

Table 3.2: Order Conditions for the RKMK method of up to order 4.

Order		RKMK Order Conditions		
1	$v(0)^{(0)} = \widehat{v}(0)^{(0)}$	$b^T \mathbf{1} = 1$		
2	$v(0)^{(1)} = \widehat{v}(0)^{(1)}$	$2b^T c = 1$	$\Rightarrow$	$b^T c = \frac{1}{2}$
3	$v(0)^{(2)} = \widehat{v}(0)^{(2)}$	$3b^T c^2 = 1$	$\Rightarrow$	$b^T c^2 = \frac{1}{3}$
		$6b^T Ac = 1$	$\Rightarrow$	$b^T Ac = \frac{1}{6}$
4	$v(0)^{(3)} = \widehat{v}(0)^{(3)}$	$4b^T c^3 = 1$	$\Rightarrow$	$b^T c^3 = \frac{1}{4}$
		$8b^T cAc = 1$	$\Rightarrow$	$b^T cAc = \frac{1}{8}$
		$12b^T Ac^2 = 1$	$\Rightarrow$	$b^T Ac^2 = \frac{1}{12}$
		$24b^T A^2 c = 1$	$\Rightarrow$	$b^T A^2 c = \frac{1}{24}$

Note that the order conditions displayed in Table 3.2 are the same as the order conditions for the classical Runge-Kutta method of the same order. The important point is that since the products on $\mathfrak{G}_s$ are commutative on the left hand part before $\otimes$, then we can derive the same order conditions that are described by the elementary differentials in the classical Butcher trees. As mentioned in the previous section, the only difference exists in the ordering of the branches in the trees. Otherwise, everything else is equivalent. Hence, the order of the RKMK method is the same as the order of the Runge-Kutta method used in the RKMK Algorithm. This is presented as the following Theorem in [27].

Theorem 3.13. *Let A_{ij}, b_j be the coefficients from a Butcher tableau for a classical Runge-Kutta method of order q. Let $c_i = \sum_{j=1}^{s} A_{ij}$. Then the RKMK Algorithm presented in Section 3.3 has order q on any manifold $\mathcal{M} = G$.*

This completes the derivations for the RKMK order conditions, and gives a way of proving that the order of a RKMK method is the same as the order of the classical Runge-Kutta method. Note that Theorem 3.13 is essentially the same as Theorem 3.10 in Section 3.3, so the discussion here with the rooted trees and bar identity operators in $\mathfrak{G}_s$ is an alternative way of proving Theorem 3.10.

CHAPTER 4

Numerical Experiments

This is a chapter on the applications of the Magnus series expansion and the Runge-Kutta-Munthe-Kaas (RKMK) methods presented in Chapter 2 and Chapter 4, to solve several ordinary matrix differential equations. Each section in this chapter features a separate system of ordinary differential equations to be solved numerically. The results from the Lie group integrators are then compared with the results from some of the classical integrators such as Runge-Kutta (RK) methods, the Gauss methods mentioned in Chapter 2, and the build-in integrator ODE45 from Matlab. ODE45 is based on an explicit Runge-Kutta (4th order, 5 stages) formula, the Dormand-Prince pair, and throughout this chapter, we will assume that ODE45 produces the most accurate results. For more information on ODE45, please refer to [10]. With the exception of ODE45, the experiments performed in this chapter will involve integrators based on fixed stepsize h and fixed order. The relevant tables of results and the Matlab code are presented in Appendix A and Appendix B.

4.1 Harmonic Oscillators

Unforced harmonic oscillators can be modelled by the following second order homogeneous differential equation [4],

$$m\frac{d^2y}{dt^2} + b\frac{dy}{dt} + ky = 0, \tag{4.1}$$

where $m, k > 0$, and $b \geq 0$. If $b = 0$, then the system is undamped. However, if $b > 0$, then different types of behaviour are possible. For the above harmonic oscillator equation, the characteristic equation is,

$$\frac{-b \pm \sqrt{b^2 - 4mk}}{2m}.$$

The three different possibilities for the roots of the characteristic equation are,

- If $b^2 - 4mk < 0$, then we have complex roots and the harmonic oscillator is said to be **underdamped**. In this case, we expect the system to oscillate about its equilibrium position.

- If $b^2 - 4mk = 0$, then we have repeated roots and the oscillator is **critically damped**.

- If $b^2 - 4mk > 0$, then the roots are real and distinct, and the oscillator is said to be **over-damped**, and the system will move to its equilibrium position without any oscillations.

We now consider the second order homogeneous differential equation given as an initial value problem, with $m = 1$, $k = 1$ and $b = 0.01t$,

$$\frac{d^2y}{dt^2} + 0.01t\frac{dy}{dt} + y = 0, \qquad y(0) = -1, \quad \dot{y}(0) = 2. \tag{4.2}$$

We note that for Eq (4.2), at time $t = 0$ the system is undamped, and the critical damping value is given by $t^* = 200$. The interval of time $(0, 200)$ and $(200, \infty)$ corresponds to the system being underdamped and overdamped respectively.

To solve Eq (4.2) with numerical integrators, we can rewrite this second order homogeneous differential equation as a system of first order differential equations, by using the substitution $dy/dt = v$, and a vector $Y(t) = [y(t), v(t)]$,

$$\dot{Y}(t) = A(t)Y(t), \tag{4.3}$$

where the matrix $A(t)$ and the initial condition $Y(0)$ are given by,

$$A(t) = \begin{bmatrix} 0 & 1 \\ -1 & -0.01t \end{bmatrix}, \qquad Y(0) = \begin{bmatrix} -1 \\ 2 \end{bmatrix}.$$

Firstly, we solved the matrix differential equation with the classical methods like the Runge-Kutta method (RK4) and Gauss methods (Gauss) discussed in Section 2.5.1. Then, we solved the same differential equation with the Magnus series expansion method, presented at the end of Section 2.5.2. All 3 integrators are of order 4, so when stepsize h is halved, their global error ratios should change roughly by a factor of 16. This can be observed from the numerical results displayed in Table A.1 in Appendix A, for the integration period from $t_0 = 0$ to $t_f = 0.1$. Also, it turns out that if we plot such results on a log-log scale of global error against stepsize h, then solutions from the Magnus method have a better accuracy in comparison with Runge-Kutta or Gauss methods of the same order. This can be observed from Figure 4.1.

Next, we turn our attention to the numerical solutions from integrating Eq (4.3) along a certain interval with fixed stepsize $h = 1/20$. We integrated the differential equation firstly over the integration period $t = [0, 40]$, with the 3 integrators RK4, Gauss, and the Magnus methods, all methods of order 4. Then we also integrate over the same period with the highly accurate ODE45 in Matlab, with a specified tolerance of 10^{-14}. Comparisons can then be done between numerical solutions from the 3 integrators under consideration, and that of Matlab's ODE45 integrator. Note that the interval $t = [0, 40]$ is below the critical damping value of $t^* = 200$ in Eq (4.3). This implies that the dynamical system is underdamped, and we should expect oscillations to occur. Similarly, we can integrate over the interval $t = [220, 230]$, which is now above $t^* = 200$. So we would not expect any oscillations in the numerical solutions since the system is overdamped in this case.

Figure 4.2 and Figure 4.3 contain the results from the experiments mentioned above. As seen from the top 3 graphs in Figure 4.2, solutions $y(t)$ are oscillating against time t throughout the integration period in the underdamped system, while the results from the overdamped system in the top 3 graphs of Figure 4.3 move toward the equilibrium at $y(t) = 0$, with no oscillations against t in the integration interval of $t = [220, 230]$.

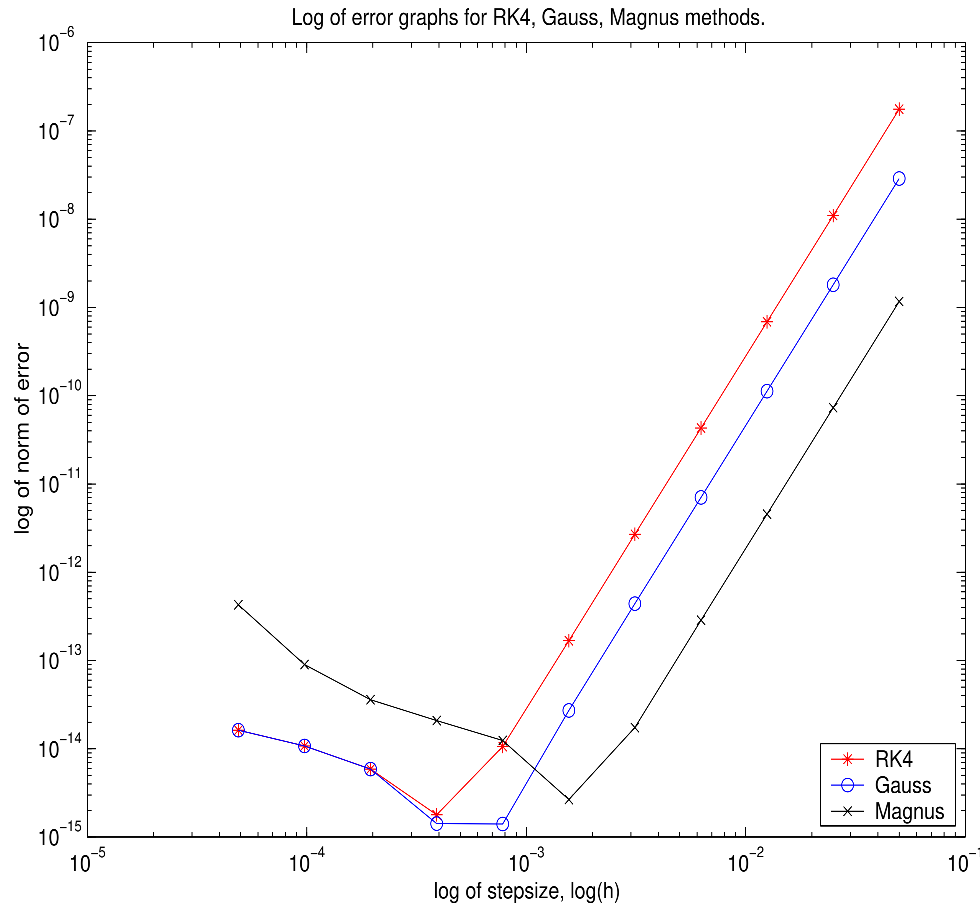


Figure 4.1: Log-log plot of global error against stepsize for results from RK, Gauss, and Magnus methods of order 4. They solved the initial value problem in Eq (4.3), from $t_0 = 0$ to $t_f = 0.1$.

Also, note that the accuracy achieved from the Magnus method is higher than that of the Runge-Kutta or the Gauss methods of the same order. This can be observed if we compare results from the 3 integrators and that of ODE45. The differences in this comparison are also plotted in the bottom 3 graphs of Figure 4.2 and Figure 4.3. For both the underdamped and overdamped systems, the difference between ODE45 and Magnus method are several degree smaller in magnitude than the differences between numerical solutions from RK4 and Gauss methods against ODE45's. With a relatively low computational cost, and a relatively good accuracy for fixed stepsize h, these brief experiments suggest the suitability of using the Magnus series expansion to integrate systems of first order differential equations describing the dynamics of an unforced harmonic oscillator.

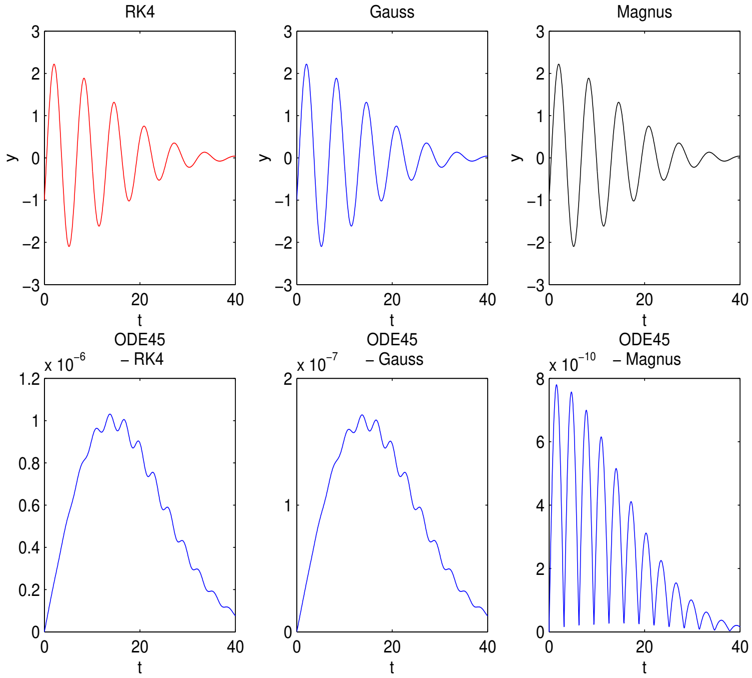


Figure 4.2: Solving the underdamped system in Eq (4.3) for the interval $t \in [0, 40]$, for which $t < t^* = 200$. The top 3 graphs are plots of numerical solutions $y(t)$ against time t, with $h = 1/20$. The bottom 3 graphs are the differences between numerical solutions from each of the 3 integrators and the highly accurate ODE45.

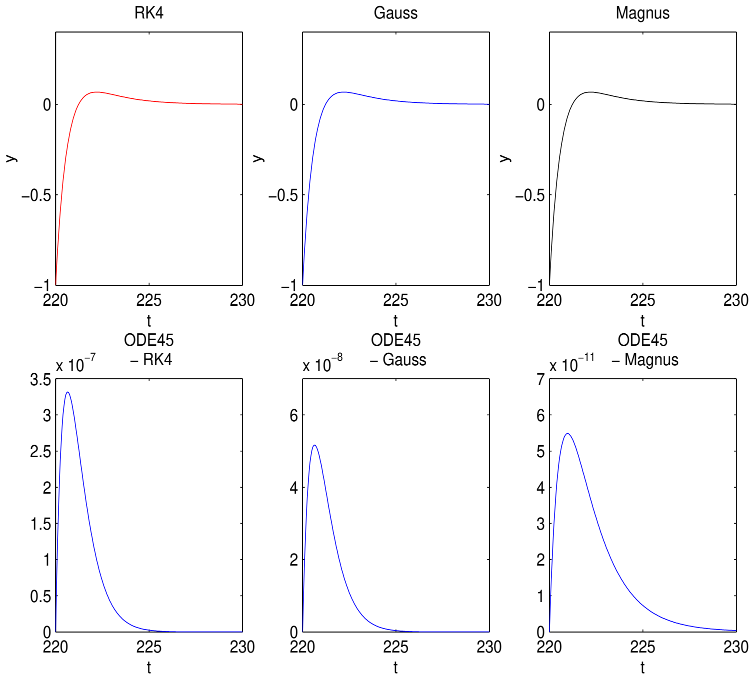


Figure 4.3: Solving the overdamped system in Eq (4.3) for the interval $t \in [220, 230]$, for which $t > t^* = 200$. The top 3 graphs are plots of numerical solutions $y(t)$ against time t, with $h = 1/20$. The bottom 3 graphs are the differences between numerical solutions from each of the 3 integrators and the highly accurate ODE45.

4.2 Lorenz Equations

The Lorenz equations describe a complex, 3 dimensional dynamical system with 3 parameters σ, ρ and β. This system has the form,

$$\begin{aligned} \dot{x} &= \sigma(y-x) \\ \dot{y} &= \rho x - y - xz \\ \dot{z} &= xy - \beta z \end{aligned} \tag{4.4}$$

and is nonlinear with two quadratic terms xy and xz. The properties of this deceptively simple system of equations turn out to be very complicated, and a good summary of results concerning the Lorenz equations is provided in [32]. However, one property of Eq (4.4) which we will make use of is the chaotic behaviour of solutions for certain parameter values.

In [32], the term **chaos** is loosely associated with the idea that long-term behaviour of solutions in a deterministic system is exhibiting sensitive dependence on initial conditions. That makes any long term integration difficult as nearby trajectories will separate exponentially fast. This behaviour makes for a basis for comparing different numerical methods, especially if we want to check the accuracy of solutions from numerical integrators against one such as ODE45 which is highly accurate. For the same stepsize h, the time it takes in the integration period for the solutions from a particular method to diverge from that of ODE45's can be the measure of how accurate that method is. That is, the longer the solutions from a particular integrator are able to follow the solutions from ODE45, then the more accurate that method is. It should be easy to identify this length of time, since solutions in the Lorenz equations are sensitive to initial conditions, and slight changes to the values of the solution in any one of the 3 components can cause drastic differences in the solutions at a later time.

Also, it turns out that the Lorenz equations are an example in a Matlab toolbox known as **DiffMan** [12]. Developed by Engø, Marthinsen, and Munthe-Kaas in Norway, DiffMan utilises the calculation and graphical powers of Matlab to solve ordinary differential equations on manifolds, using a variety of Lie group techniques. In the examples for solving the Lorenz equations, it is interesting to note that when using the RKMK method, the manifold $\mathcal{M}$ is defined as the domain of groups acting on themselves, the Lie group G is simply $\mathbb{R}^n$, and the Lie algebra is also $\mathbb{R}^n$ and abelian. The Lie algebra action λ is given by the standard vector addition, $\lambda(v,y) = v + y$ for v and $y \in \mathbb{R}^n$, and so the mapping $f : \mathbb{R} \times \mathcal{M} \to \mathfrak{g}$ in Assumption 1.1 is simply $f : \mathbb{R}^n \to \mathbb{R}^n$ for this setting, and the generic presentation of differential equations (or the infinitesimal generator $v_{\mathcal{M}}$) is reduced back to the original differential equation, where f and F are now the same map [28],

$$\dot{y}(t) = F(y), \qquad \text{where } F : \mathbb{R}^n \to \mathbb{R}^n.$$

In other words, the Lorenz equations can be solved by the RKMK method in its present form in Eq (4.4). Also, because the Lie algebra is abelian, the adjoint operator $\mathrm{ad}_u^0(v) = v$ and $\mathrm{ad}_u^n(v) = 0$ for $n \geq 1$. Then, the inverse of the differential of the exponential map $\mathrm{dexp}_u^{-1}(v) = v$, and in the RKMK algorithm given in Section 3.3, $\widetilde{k}_i = k_i$. Hence, **the entire RKMK algorithm is reduced to the classical Runge-Kutta method**.

Now, let us follow the approach of Lorenz used when he numerically solved the Lorenz equations to study the behaviour of the trajectories for a given initial condition $Y(t) = (x(t), y(t), z(t)) \in \mathbb{R}^3$. He studied the particular case when $\sigma = 10$, $\beta = 8/3$, and $\rho = 28$ [32]. We set the initial condition $Y(t)$ as $Y(0) = (17, -21, 54)$, and integrate over the interval $t = [0, 10]$ with a constant stepsize of $h = 1/20$. The integrators we used for solving this example include the classical 4th order RK method, the implicit 4th order Gauss method, the RKMK method (which is the same as the RK method in this case), and finally, Matlab's ODE45 for comparison.

Plotting the results in three dimensions in Figure 4.4, we see the appearance of the famous **Lorenz butterfly** from all four of the integrators used to solve Eq (4.4). This is known as the chaotic attractor, which is loosely defined as a set A to which all neighbouring trajectories converge. A is a closed set, such that A is an invariant set, and attracts an open set of initial conditions within the basin of attraction of A [32].

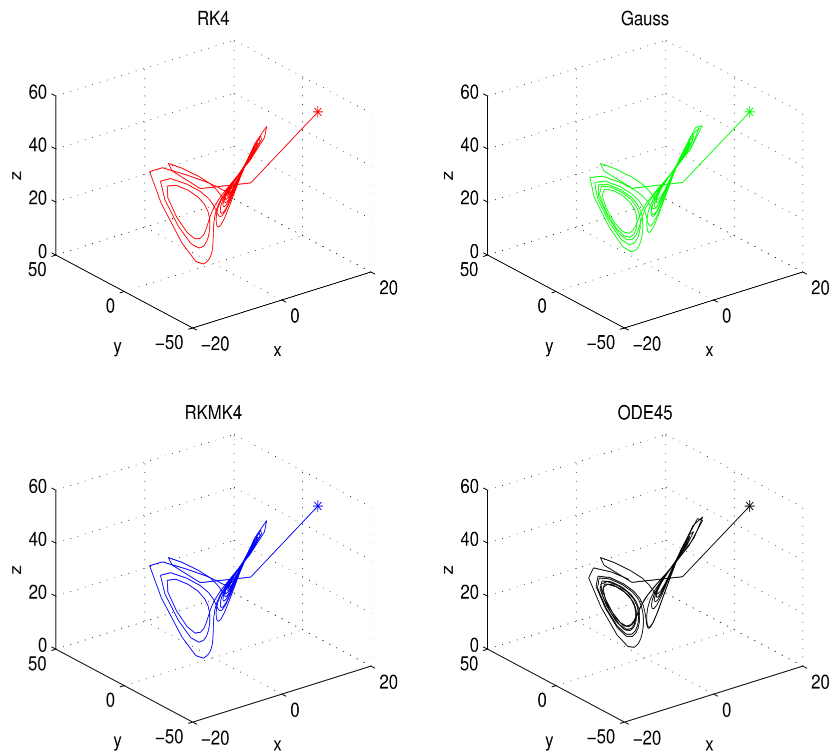


Figure 4.4: 3 dimensional graphs showing the Lorenz attractors from 4 different integrators, when solving Eq (4.4). $y_0 = [17, -21, 54]$, $t_0 = 0$, $t_f = 10$, and $h = 1/20$.

Although all four attractors have the same form in three dimensions, the behaviour of trajectories from different methods is not the same. This can be seen from the plots of the three components of solutions $Y(t)$ from each of the four integrators tested, against the integration time interval $t = [0, 10]$ in Figure 4.5. Note that the three component solutions from both RK4 and RKMK4 are exactly the same, as we expected from the discussion previously. Assuming that ODE45 method produced the most accurate solutions, then we can see that the behaviour of solutions from other integrators do not follow that of ODE45's for the entire integration period. This is due to the sensitivity of solutions to the initial conditions, and in this case, the value of Y_i at the beginning of the ith integration step. For all three components $x(t), y(t), z(t)$, the solutions from RK/RKMK, and Gauss methods are pretty close to the solutions from ODE45 from the initial value $t = 0$ to roughly $t = 3$. However, after that, solutions from RK/RKMK methods slowly drifted away from the ODE45's trajectories until their behaviours are completely different from roughly $t = 5.5$. Similarly, the solutions from the Gauss method are slightly better than those from RK/RKMK methods, in the sense that the three components solutions from the Gauss method followed ODE45's trajectories for a longer length of time, until they drifted apart at roughly $t = 6$. In this case, of the three fixed stepsize integrators the Gauss method performs better than the RK/RKMK methods.

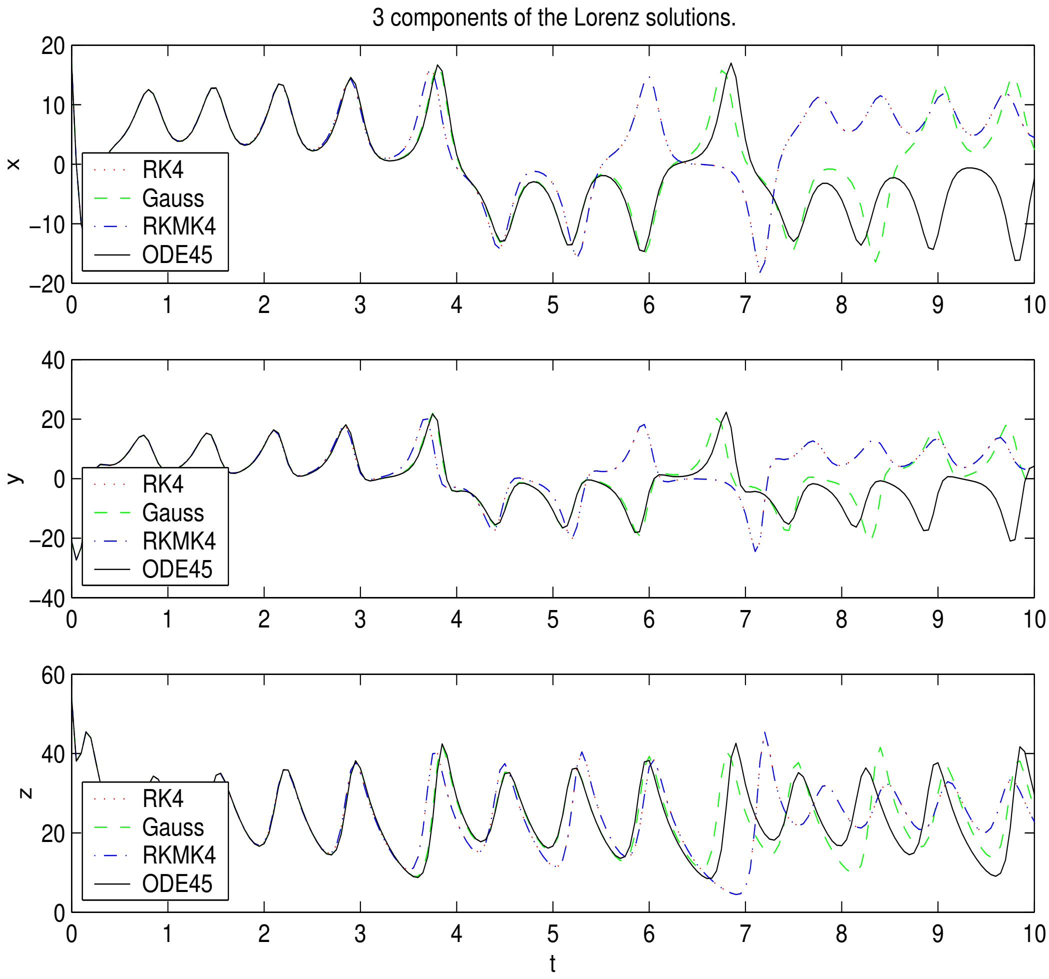


Figure 4.5: 3 component solutions $Y(t) = (x(t), y(t), z(t))$ against time t, from solving the Lorenz Equations in Eq (4.4). $y_0 = [17, -21, 54]$, $t_0 = 0$, $t_f = 10$, and $h = 1/20$.

4.3 Special Orthogonal Matrix Differential Equations

This next experiment is designed to fit into the framework of the RKMK method, and is perhaps one of the most artificial applications of all. This is because the matrix differential equation to be solved is designed to feature the advantage of using the RKMK method, which preserves some properties of the solution in the system. In this case, we will be working with the matrix Lie group SO(3) and its associated Lie algebra $\mathfrak{so}(3)$, while the characteristic we want to preserve in the numerical solution is the orthogonality of the matrix $y(t) \in \mathrm{SO}(3)$. Let us consider $\mathcal{M} = G = \mathrm{SO}(3)$, and use the exponential map as our coordinate mapping from $\mathfrak{so}(3)$ to SO(3). Then, the Lie algebra action we want to use is simply the matrix multiplication,

$$\lambda(v, p) = \Lambda(\exp(v), p) = \exp(v) \cdot p, \qquad \text{where} \quad v \in \mathfrak{so}(3), \quad p \in \mathcal{M} = \mathrm{SO}(3).$$

The autonomous system of differential equations we want to solve is given by,

$$\dot{y}(t) = f(y(t))y(t), \qquad y(0) = y_0, \tag{4.5}$$

where $y(t) \in \mathrm{SO}(3)$, and $f : \mathrm{SO}(3) \to \mathfrak{so}(3)$ and has the form in Matlab [28],

$$f(y) = \mathrm{diag}(\mathrm{diag}(y, 1), 1) - \mathrm{diag}(\mathrm{diag}(y, 1), -1).$$

This function $f(y)$ produces a skew-symmetric matrix in the corresponding Lie algebra $\mathfrak{so}(3)$. The initial condition y_0 was generated by finding the matrix Q in the QR method for a random 3×3 matrix in Matlab, which has the following values,

$$y_0 = \begin{bmatrix} -0.30629995334276 & -0.46674863567359 & -0.82965417474937 \\ -0.06581027265024 & -0.85907704683087 & 0.50759790742481 \\ -0.94965748909590 & 0.21007698280677 & 0.23241840438933 \end{bmatrix}.$$

Given that we know $y(0) = y_0$, we want to solve this system to find the matrix solution $y(t_f)$ for some time in the future t_f.

4.3.1 Matrix Exponential and the Padé Approximations

One very important relationship between elements in the Lie group G and its corresponding Lie algebra $\mathfrak{g}$ is the exponential map. For most case, especially in numerical experiments, we are dealing with matrix Lie group and matrix Lie algebra. Then, as mentioned in Chapter 1, the matrix exponential $\exp : \mathfrak{g} \to G$ is given by the sum,

$$\exp(v) = \sum_{k=0}^{\infty} \frac{v^k}{k!}, \quad \forall v \in \mathfrak{g}.$$

In numerical computations, the mapping Ψ used in the approximate the matrix exponential must be one such that $\Psi : \mathfrak{g} \to G$. This allows for the calculation of Ψ to be of the order of the integrator rather than of machine precision [19]. Because this operation is important in the RKMK Algorithm, we need to find ways of approximating this matrix exponential accurately enough, such that it would not destroy the accuracy achieved for the solutions in the RKMK methods.

As mentioned in Chapter 3, if the Lie group G is quadratic, then the Cayley map or the diagonal Padé approximants can be used to approximate the matrix exponential map. In Moler and Van Loan's paper [25], they listed nineteen "dubious" ways of finding the exponential of a matrix, of which the method of Padé approximation was included. The discussion was motivated by the knowledge that the solution $x(t)$ for a matrix differential equation, $\dot{x}(t) = Ax(t)$, has the form, $x(t) = e^{tA}x_0$, where e^{tA} is defined by the convergent power series,

$$e^{tA} = \mathrm{Id} + tA + \frac{(tA)^2}{2!} + \cdots,$$

where Id is the identity matrix. Given that for our particular ODE, the matrix norm of the solution y is usually small, then the method by Padé Approximation is likely the better choice from the range of methods discussed in [25].

The (p, q) Padé approximation to e^A is defined by $R_{p,q} = [D_{p,q}(A)]^{-1}N_{p,q}(A)$, where,

$$\begin{aligned} N_{p,q}(A) &= \sum_{k=0}^{p} \frac{(p+q-k)!p!}{(p+q)!k!(p-k)!}A^k, \\ D_{p,q}(A) &= \sum_{k=0}^{q} \frac{(p+q-k)!q!}{(p+q)!k!(q-k)!}(-A)^k. \end{aligned}$$

In *Method 2* of [25], some of the issues concerning the use of the Padé approximations to find e^A are discussed in detail. The main points which are relevant to our solving of the ODE in Eq (4.5) include,

- Singularity of $D_{p,q}$ is avoided if p and q are chosen to be large enough.
- Roundoff error makes the Padé approximations unreliable.
- Padé approximation can be used if $\|A\|$ is not too large.
- The diagonal approximant $(p = q)$ is preferred over the off diagonal approximant $(p \neq q)$.

To overcome some of the problems associated with the Padé approximations, the following fundamental property of the exponential function is used,

$$e^A = (e^{A/m})^m,$$

where m is chosen to be a power of two, often given by the smallest power of two for which $\|A\|/m \leq 1$. The actual result for e^A is recovered by repeated squaring. This method is considered to be one of the most effective way of finding the matrix exponential, and the implementation of **expm** in Matlab is based on this scaling and squaring Padé Approximation technique, given in *Algorithm 11.3.1* from [22]. The Padé approximant used in **expm** is the diagonal approximant $R(6, 6)$,

$$R(6,6) = \frac{\mathrm{Id} + \frac{1}{2}A + \frac{5}{44}A^2 + \frac{1}{66}A^3 + \frac{1}{792}A^4 + \frac{1}{15840}A^5 + \frac{1}{665280}A^6}{\mathrm{Id} - \frac{1}{2}A + \frac{5}{44}A^2 - \frac{1}{66}A^3 + \frac{1}{792}A^4 - \frac{1}{15840}A^5 + \frac{1}{665280}A^6},$$

and the coefficient for each power of the matrix A^k is a scalar multiple of the coefficient from the previous power of A. (i.e. coefficient c_k for A^k is given by $c_k = zc_{k-1}$, where $z \in \mathbb{R}$.) The ratio, z, is given by,

$$z = \frac{c_k}{c_{k-1}} = \left[\frac{(2q-k)!q!}{(2q)!k!(q-k)!}\right] / \left[\frac{(2q-(k-1))!q!}{(2q)!(k-1)!(q-(k-1))!}\right] = \frac{q-k+1}{k(2q-k+1)}.$$

Matlab's built-in function **expm** is coded in C, but the same algorithm is also implemented in Matlab code as **expm1.m**. For further details please refer to Appendix B.

Given that we have briefly discussed the matrix exponential calculation, and the details of the RKMK Algorithm in Chapter 3, we now want to apply this technique to solve the differential equation in Eq (4.5).

4.3.2 RK4 and RKMK4 Results

Given the ordinary differential equations in Eq (4.5) and the initial conditions y_0 in Section 4.3, we want to solve for the special orthogonal matrix $y(t)$ using both the traditional RK method as well as the RKMK method introduced in Chapter 3.

Firstly, we want to check that the RKMK method preserves the order q of the RK method on which it is based. To do this, the problem in Eq (4.5) was integrated from $t_0 = 0$ to $t_f = 0.1$, with both explicit RK and RKMK methods, based on three different Butcher Tableau in [6], with number of stages and orders $s = 2, q = 2$, and $s = 3, q = 3$, and $s = 4, q = 4$ respectively from left to right.

$$
\begin{array}{c|cc}
0 & & \\
\frac{1}{2} & \frac{1}{2} & \\
\hline
 & 0 & 1
\end{array}
\qquad
\begin{array}{c|ccc}
0 & & & \\
\frac{2}{3} & \frac{2}{3} & & \\
0 & -\frac{3}{2} & \frac{3}{2} & \\
\hline
 & \frac{1}{12} & \frac{3}{4} & \frac{1}{6}
\end{array}
\qquad
\begin{array}{c|cccc}
0 & & & & \\
\frac{1}{2} & \frac{1}{2} & & & \\
\frac{1}{2} & 0 & \frac{1}{2} & & \\
1 & 0 & 0 & 1 & \\
\hline
 & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6}
\end{array}
$$

Figure 4.6 displays the natural logarithm of the global errors from the different methods against stepsize h. The global error $\|y_i - y_{i-1}\|$ corresponding to each halving of stepsize h_i was calculated from the 2-norm of the difference between the final solution y_i and y_{i-1} at the previous h_{i-1}. We observe that the slopes of the errors generated by the three RKMK methods are roughly parallel to those from the corresponding RK methods. This implies that the structure of the RKMK method preserves the order q of the underlying RK method, as stated by Theorem 3.10 in Section 3.3. In the case of the solutions y from 2nd and 3rd order methods for solving Eq (4.5), RKMK2 and RKMK3 produced slightly more accuracy than those from the traditional RK2 and RK3 methods. This effect might be due to the preservation of orthogonality in the solution matrix y from RKMK methods. However, the reverse is true for the comparison between RK4 and RKMK4 methods. One contributing factor might be the greater number of evaluations for each step of the RKMK method than the RK method, and hence a greater accumulation of round-off error exists in the solutions from the RKMK method. This build up of round-off errors might be large enough to overcome the accuracy achieved by preserving the orthogonality of matrix y, so that in the comparison, results from RKMK4 are slightly less accurate than those from RK4.

As mentioned at the beginning of Section 4.3, since the solution $y \in \mathrm{SO}(3)$, the two properties we want to preserve in the numerical solutions when solving Eq (4.5) are the orthogonality and unit determinant of matrix y. In Figure 4.7, two plots are shown. Both RK and RKMK methods of order 4 are used. The left hand side graph shows the errors in the unit determinant from the matrix $y(t)$ against the integration period $t \in [0,\ 10]$. The right hand side graph is the error in the orthogonality of $y(t)$ against the same time interval t.

From both of the graphs in Figure 4.7, the RKMK method is able to preserve the orthogonality and unit determinant, but errors of magnitude 10^{-7} exist for solutions with respect to both properties, from the traditional RK method using a stepsize of $h = 1/10$. In this case, the RKMK method is considered as the geometric integrator for the solving of the ODE in Eq (4.5).

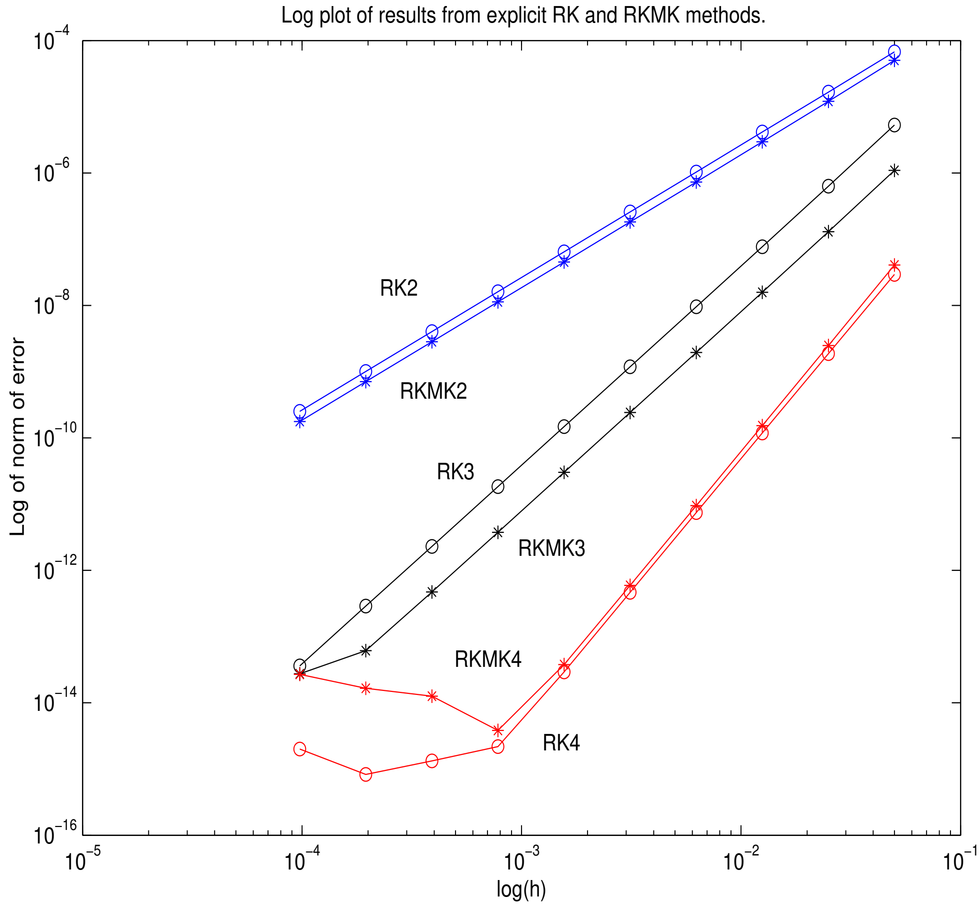


Figure 4.6: Log-log plot of global error against stepsize for results from RK and RKMK methods with orders $q = 2, 3, 4$, from solving the initial value problem in Eq (4.5). $t_0 = 0$ and $t_f = 0.1$.

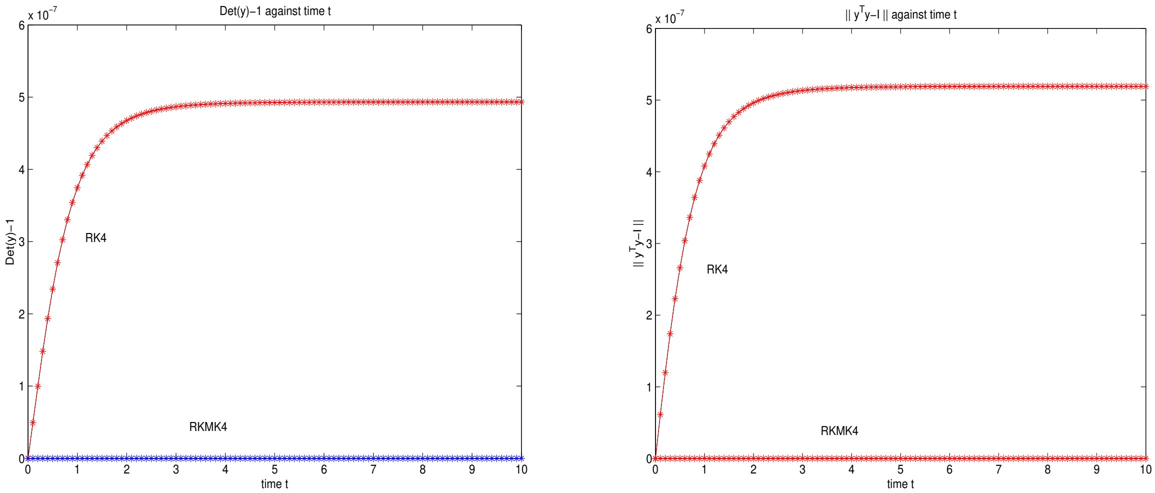


Figure 4.7: Solutions from RK and RKMK methods of $q = 4$, when solving Eq (4.5). The left hand side graph is the error in $\det(y)$. The right hand side graph is the error in the orthogonality of y, given by the 2-norm $\|y^T y - I\|$. $t_0 = 0$, $t_f = 10$ and $h = 1/10$.

4.3.3 Compensated Error Addition for RK method

The motivation behind the compensated error addition is to reduce the round-off error which is accumulated when the calculations have reached machine precision. The round-off error will dominate all calculations when we are dealing with numbers close to the magnitude of 10^{-15}.

The idea of this compensated error addition (or otherwise known as the summation of a large number of small numbers), first appeared in the Gill-Møller Algorithm, discussed in [7]. Suppose that we start with the traditional RK method. Let y_0 be the solution at the beginning of the step. Then a new step in the RK method is given by $y_1 = y_0 + \delta$, where δ is a linear combination of function evaluations which is dependent on stepsize h. If h is small, then the multiplication of a small number by another will cause relevant information to be pushed beyond machine precision. However, if this error, denoted z, can be estimated, then we can add z back to δ in the next step as a compensation for the rounding error created in the previous step.

One way to estimate this rounding error z, is to take the difference between δ and $y_1 - y_0$ [7]. Note that δ denotes the exact amount which should be added to y_0 after one numerical step, and calculated with a reasonable amount of accuracy, while the difference $y_1 - y_0$ is the actual amount containing the rounding error, that the computer has added to y_0 instead. Hence, similar to the example of Euler method given in [7], we can slightly modify the traditional RK method with the compensated error addition technique to estimate the rounding error z.

RK method with compensated error addition

From t_0 to t_1, one step of the RK method with compensated error addition on an autonomous function $F(y)$ is given by the following scheme:

> Let a_{ij}, b_j, and c_i be the coefficients of an s-stage, q'th order classical Runge-Kutta method, and let $c_i = \sum_{j=1}^{s} a_{i,j}$. Let z be the rounding error estimate, δ is the small change to the original step y_0, and h is the stepsize.
>
> Given t_0, y_0, z_0
> for $i = 1, 2, ..., s$
> $\quad k_i = F(y_0 + h \sum_{j=1}^{s} a_{i,j} k_j)$
> end
> $\delta = h \sum_{i=1}^{s} b_i k_i + z_0$
> $t_1 = t_0 + h$
> $y_1 = y_0 + \delta$
> $z_1 = \delta - (y_1 - y_0)$
> Output t_1, y_1, and z_1.

Note that at the beginning of the integration period, the rounding error estimate z_0 is usually set to 0. In the subsequent steps, z_0 is replaced by the rounding error matrix from the previous step. Also, if the function to be integrated is non-autonomous $F(t, y)$, then a subtraction of $h - (t_1 - t_0)$ must be done for the independent variable t as well, to compensate for the rounding error there.

So now we want to integrate Eq (4.5), with the initial matrix y_0 given in Section 4.3. With this particular matrix y_0, its determinant is $\det(y_0) = 1$ to 15 decimal places, and $\|y(0)^T y(0) - \mathrm{Id}\| = 1.735689226962107e^{-14}$. Note that because the matrix y_0 is created from QR method in Matlab from a random 3×3 matrix, then the matrix y_0 is created to be orthogonal only to machine precision. We integrated from $t_0 = 0$ to $t_f = 10$. The results, from both the original RK method and the same RK method with compensated error addition, are displayed in Figure 4.8. The numerical values can be found in Table A.2 and Table A.3 in Appendix A.

Figure 4.8 is a log-log graph of the global error against halving of the stepsize h for both the traditional RK method, and the modified RK method with compensated error addition. The calculations done here are similar to those in Section 4.3.2. The 'ratio' columns in Table A.2 and Table A.3 are found by taking the ratio of the left hand column $\|y_i - y_{i-1}\|/\|y_{i-1} - y_{i-2}\|$. As shown in the plot, at the stepsize for which error in the solutions are close to 10^{-15}, the rounding error began to dominate the calculations, and results beyond this point become unreliable. However, the effect of this rounding error can be temporarily overcome, if the compensated error technique is applied to the calculations. This technique improved on the accuracy of the solution, especially for stepsize h when the rounding error became significant for the original RK method.

In more details, if we compare Table A.2 with Table A.3, the compensated error technique increased the accuracy of the solutions roughly by 3 more decimal places. This can be observed if we compare the global error columns between the two tables, especially for results from stepsize smaller than $h = 1/640$ from both methods, where rounding error started to dominate the solutions from the original RK method.

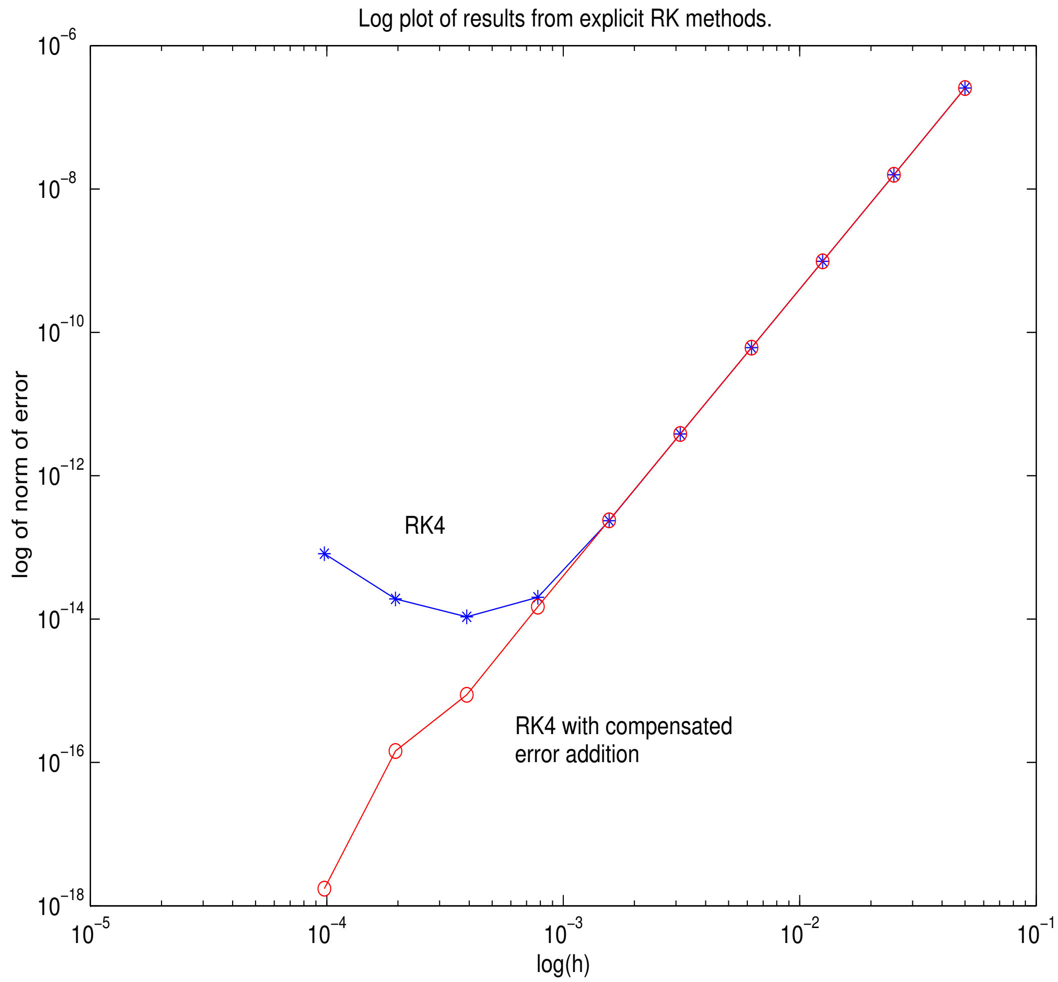


Figure 4.8: Log-log plot of global error against stepsize for results from the original RK method, and RK method with compensated error addition. Both integrators solved the initial value problem given in Eq (4.5), from $t_0 = 0$ to $t_f = 10$.

Nevertheless, even with this increased accuracy from the compensated error technique, the unit determinant and orthogonality of solution y are destroyed. This is as expected since the classical method such as RK method is not designed to preserve such properties in the numerical solution. Hence, we want to apply the same compensated error technique to the RKMK method, and see whether this will improve the accuracy in the solutions from the Lie group integrator.

4.3.4 Compensated Error Addition for RKMK method

We want to apply the same idea of estimating the round-off error z as in the previous section to the structure in the RKMK method. One major difference between the traditional RK method and the RKMK method is that the operation in the internal stages of an RKMK method is given by the Lie algebra action $\lambda : \mathfrak{g} \times \mathcal{M} \to \mathcal{M}$. Since the matrix exponential is involved in the RKMK algorithm, then the implementation of the compensated error technique can no longer be simply the addition of the estimated rounding error z to the current y value at the beginning of the integration step.

However, we can still estimate the rounding error in a slightly different way. Suppose that we use the exponential map as the coordinate mapping in the RKMK method, and that in Matlab we use its matrix exponential **expm** to map from $\mathfrak{g} = \mathfrak{so}(3)$ to $G = \mathrm{SO}(3)$. Given that the matrix $v \in \mathfrak{so}(3)$ for this initial value problem in Section 4.3 has a small norm, then $\mathrm{expm}(v)$ should be close to the identity matrix Id. Knowing that the final step in a RKMK method is found by the Lie algebra action,

$$y_1 = \lambda(v, y_0) = \mathrm{expm}(v)y_0, \quad v \in \mathfrak{g},\, y_0, y_1 \in G,$$

we can rewrite this so that the final step in the RKMK method is in the classical additive form,

$$\begin{aligned} \delta &= (\exp(v) - \mathrm{Id})y_0, && (4.6)\\ y_1 &= y_0 + \delta. && (4.7) \end{aligned}$$

If δ is sufficiently small, then the rounding error z can be estimated by taking the difference, similar to the compensated error scheme for traditional method,

$$z = \delta - (y_1 - y_0). \tag{4.8}$$

At this point, several attempts were made to estimate z for the problem given in Eq (4.5), again from $t_0 = 0$ to $t_f = 10$. One of the first attempts was to modify the RKMK method in a similar way to the modification done in RK method with compensated error addition. In one step of the RKMK method, after the internal stages u_i, k_i, and $\widetilde{k}_i$ are found as in the **RKMK Algorithm** given in Section 3.3, then the following was done,

$$\begin{aligned} &v = h\textstyle\sum_{j=1}^{s} b_j\widetilde{k}_j\\ &\delta = (\mathrm{expm}(v) - \mathrm{Id})y_0 + z_0\\ &y_1 = y_0 + \delta\\ &z = \delta - (y_1 - y_0). \end{aligned}$$

Note that z_0 is the round-off error matrix from the previous step, matrix $v \in \mathfrak{so}(3)$, and matrix $y_0 \in \mathrm{SO}(3)$.

Figure 4.9 displays the log-log plot of the global error for solutions from the numerical integrators when stepsize h was halved. Note that the gain in this modification of the original RKMK method is not significant enough to overcome the effect of rounding errors in the calculations. This is due to the relatively big rounding error created in the actual operation of

$\exp(v) - \mathrm{Id}$. Even though this $\exp(v)$ is close to Id, yet this difference is still large with respect to machine precision. As a result, all information beyond the 15 digits is sacrificed. So the terms δ as well as $y_1 - y_0$ both suffer from rounding errors, and z in this calculation is not a good error estimate. Hence, another way of calculating δ must be found, in order to produce a better result in this compensated error scheme for the RKMK method.

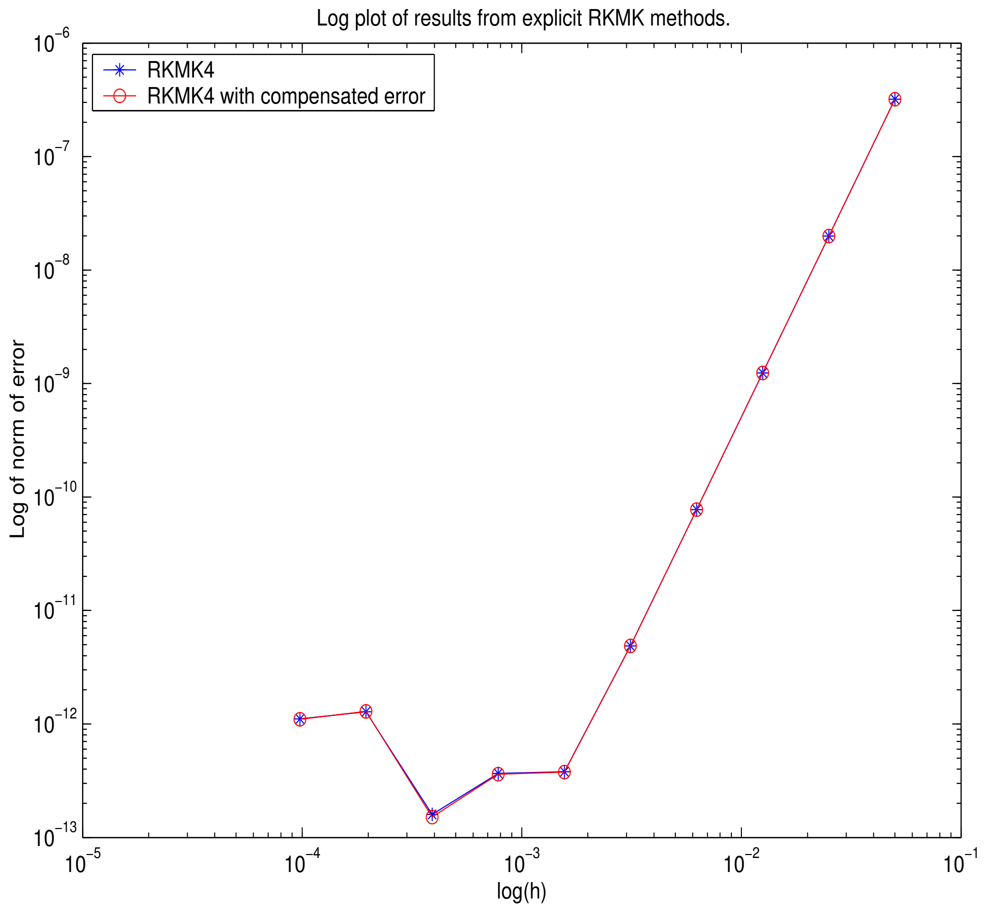


Figure 4.9: Log-log plot of global error against stepsize for results from the original RKMK method, and RKMK method with compensated error using the *first attempt*. Both integrators solved the initial value problem in Eq (4.5), from $t_0 = 0$ to $t_f = 10$.

Another attempt we made was similar to the previous example, but instead of relying on the built-in function **expm** in Matlab, this attempt uses the algebraic manipulation of the expression $\exp(v) - \mathrm{Id}$ when the matrix exponential is found by the diagonal Padé approximant $R(6,6)$, given in Section 4.3.1. So, if we use the Padé approximant $R(6,6)$ to find $\exp(A)$, then we have the following as the expression for $\exp(A) - I$,

$$\exp(A) - \mathrm{Id} = \frac{A + \frac{1}{33}A^3 + \frac{1}{7920}A^5}{\mathrm{Id} - \frac{1}{2}A + \frac{5}{44}A^2 - \frac{1}{66}A^3 + \frac{1}{792}A^4 - \frac{1}{15840}A^5 + \frac{1}{665280}A^6} \tag{4.9}$$

Similar to the technique mentioned in Section 4.3.1 for the matrix exponential, we can do a 'squaring and scaling' operation to matrix $v \in \mathfrak{so}(3)$ such that we can find a good approximation to $\exp(v) - \mathrm{Id}$. This can be done in the following way, which is a slight modification to the algorithm given in Algorithm 11.3.1 in [22], and implemented in the Matlab program **expm1.m**.

Squaring and Scaling Algorithm for $\exp(v) - \mathrm{Id}$
The following algorithm is to find $\exp(v) - \mathrm{Id}$ via the algebraic manipulation of the Padé approximant $R(6,6)$.

Scaling of the original matrix v.
$s = \max(0, 1 + \mathrm{floor}(\log_2(\|v\|_\infty)))$
$v = v/2^s$.
Algebraic manipulation of R(6,6)-Id.
$w = \frac{v+\frac{1}{33}v^3+\frac{1}{7920}v^5}{\mathrm{Id}-\frac{1}{2}v+\frac{5}{44}v^2-\frac{1}{66}v^3+\frac{1}{792}v^4-\frac{1}{15840}v^5+\frac{1}{665280}v^6}$.
Squaring of the expression.
if s is 0,
then output w.
else,
for i from 1 to s,
$w = w * w + 2 * w$
end
output w.
end

Note that the above algorithm correctly gives $\exp(v) - \mathrm{Id}$ because for $w = \exp(v/2) - \mathrm{Id}$,

$$(\exp(v/2) - \mathrm{Id})(\exp(v/2) - \mathrm{Id}) + 2(\exp(v/2) - \mathrm{Id}) = \exp(v) - \mathrm{Id}.$$

For $w = \exp(v/4) - \mathrm{Id}$,

$$(\exp(v/4) - \mathrm{Id})(\exp(v/4) - \mathrm{Id}) + 2(\exp(v/4) - \mathrm{Id}) = \exp(v/2) - \mathrm{Id}\cdots$$

Hence, putting everything together, we can implement the one step of the RKMK method with compensated error addition in the following way, for solving Eq (4.5) in Section 4.3 from $t_0 = 0$ to $t_f = 10$.

RKMK method with compensated error addition
From $t_0 = 0$ to $t_f = h$, one step of the RKMK Algorithm with compensated error addition is given by the following scheme :

Let $a_{i,j}$ and b_j be the coefficients of an s-stage, q'th order classical Runge-Kutta method and let $c_i = \sum_{j=1}^{s} a_{i,j}$. Let z be the rounding error estimate, δ be the small change to the original step y_0, and h is the stepsize.
Given t_0, y_0, and z_0,
for $i = 1, 2, ..., s$
$u_i = h\sum_{j=1}^{s} a_{i,j}\widetilde{k}_j$
$k_i = f(hc_i, \lambda(u_i, y_0))$
$\widetilde{k}_i = \mathrm{dexpinv}(u_i, k_i, q)$
end
$v = h\sum_{j=1}^{s} b_j\widetilde{k}_j$
Find w from **Squaring and Scaling Algorithm for $\exp(v) - \mathrm{Id}$**.
$\delta = w * y_0 + z_0$
$t_1 = t_0 + h$
$y_1 = y_0 + \delta$
$z_1 = \delta - (y_1 - y_0)$.
Output t_1, y_1, and z_1.

Note that $f : \mathbb{R} \times \mathcal{M} \to \mathfrak{g}$, and the Lie algebra action is $\lambda : \mathfrak{g} \times \mathcal{M} \to \mathcal{M}$.

The results from solving Eq (4.5) with the RKMK method and RKMK method with the compensated error technique, are displayed in Figure 4.10, with numerical values in Table A.4 and Table A.5 in Appendix A. As Figure 4.10 shows, this method behaves in a similar fashion to the compensated error estimate for RK method in Figure 4.8. This modified RKMK method produced solutions which are still reliable for stepsize h smaller than $h = 1/320$, which is the point when the rounding error started to dominate the solutions from the original RKMK method.

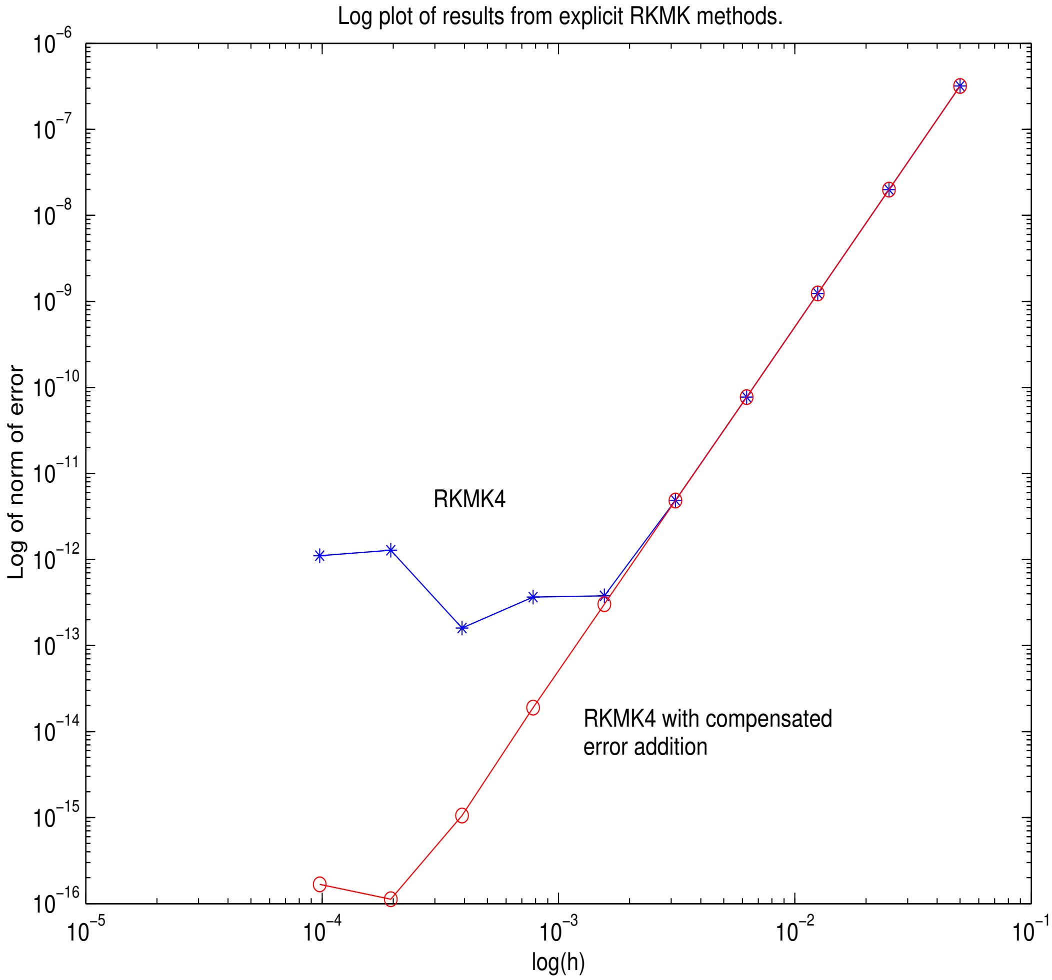


Figure 4.10: Log-log plot of global error against stepsize for results from the original RKMK method, and RKMK method with compensated error addition. Both integrators solved the initial value problem in Eq (4.5), from $t_0 = 0$ to $t_f = 10$.

4.4 Differential Equation on the Sphere

This is a differential equation designed to be solved by the Magnus series expansions and the RKMK methods, and it is another example featured in DiffMan [12].

$$\frac{dy}{dt} = \begin{bmatrix} 0 & t & -0.4\cos(t) \\ -t & 0 & 0.1t \\ 0.4\cos(t) & -0.1t & 0 \end{bmatrix} y(t) = A(t)y(t), \qquad y(0) = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}. \tag{4.10}$$

The solution $y(t) \in \mathbb{R}^3$ in Eq (4.10) is a vector of unit length, evolving on the unit sphere. That is, the solution vector $y(t)$ rotates around the spherical surface with respect to its origin, and naturally, the homogeneous manifold in this case is the unit sphere. The main difference between this example and the previous example in Section 4.3, is that the map $f : \mathbb{R} \times \mathcal{M} \to \mathfrak{g}$ in the generic presentation in Section 1.3.5, is here given by the matrix $A(t) : \mathbb{R} \to \mathfrak{so}(3)$. Whereas in the previous example, $f : \mathrm{SO}(3) \to \mathfrak{so}(3)$. Note that from the differential equation, the Lie algebra is defined by the matrix $A(t)$, which in this case is $\mathfrak{g} = \mathfrak{so}(3)$. Then, the generic presentation of the differential equation (or infinitesimal generator $v_{\mathcal{M}}$), with which we want to solve for the solution $y(t)$ is given by,

$$\dot{Q}(t) = A(t)Q(t), \qquad Q(0) = \mathrm{Id},$$

where $Q(t)$ is the corresponding 3×3 matrix in the Lie group SO(3). This is exactly the form of the differential equation we solved for in the last example using the RKMK method.

The Lie group action in this case is given by $\Lambda(v, Q) = \exp(v) \cdot Q$, producing a special orthogonal matrix, which we then use as the input matrix Q at the beginning of the next step in the integration process. At the end of each integration step, we find the position of vector $y(t) \in \mathbb{R}^3$ by multiplying the matrix $Q(t) \in \mathrm{SO}(3)$ to the initial position $y(0) \in \mathbb{R}^3$, for a rotation at the current step from the initial position at the beginning of the integration period. Hence, we have a sequence of position vectors $y(t)$ with respect to the initial position $y(0)$, which traces out the trajectories on the surface of the unit sphere at each time step of the integration process.

To perform the numerical experiments, let us suppose that we want to integrate Eq (4.10) with the given initial condition $y(0)$, over the time interval $t = [0, 64]$ with a constant stepsize of $h = 1/20$, and order $p = 4$ for the 4 integrators RK, Gauss, Magnus and RKMK methods. Once again, we also integrate the same problem with Matlab's ODE45 for comparison.

In the top four graphs in Figure 4.11, we plotted the three dimensional solutions $y(t)$ over the integration period, on the unit sphere for the four methods under consideration, using the same stepsize h. It is apparent that solutions from the RK integrator eventually drifts away from the surface of the sphere, while solutions from the other three integrators appear to adhere to the same surface. Again assuming that ODE45 produces the most accurate solutions $y(t)$ when solving the same initial value problem, then we can take a look at the difference of solutions between ODE45's and the other four integrators. Such results are plotted in the bottom four graphs in Figure 4.11. Note that the difference between ODE45's and the two Lie group methods such as the Magnus series expansion and the RKMK method are significantly smaller, than the differences between ODE45's and the classical RK and Gauss methods.

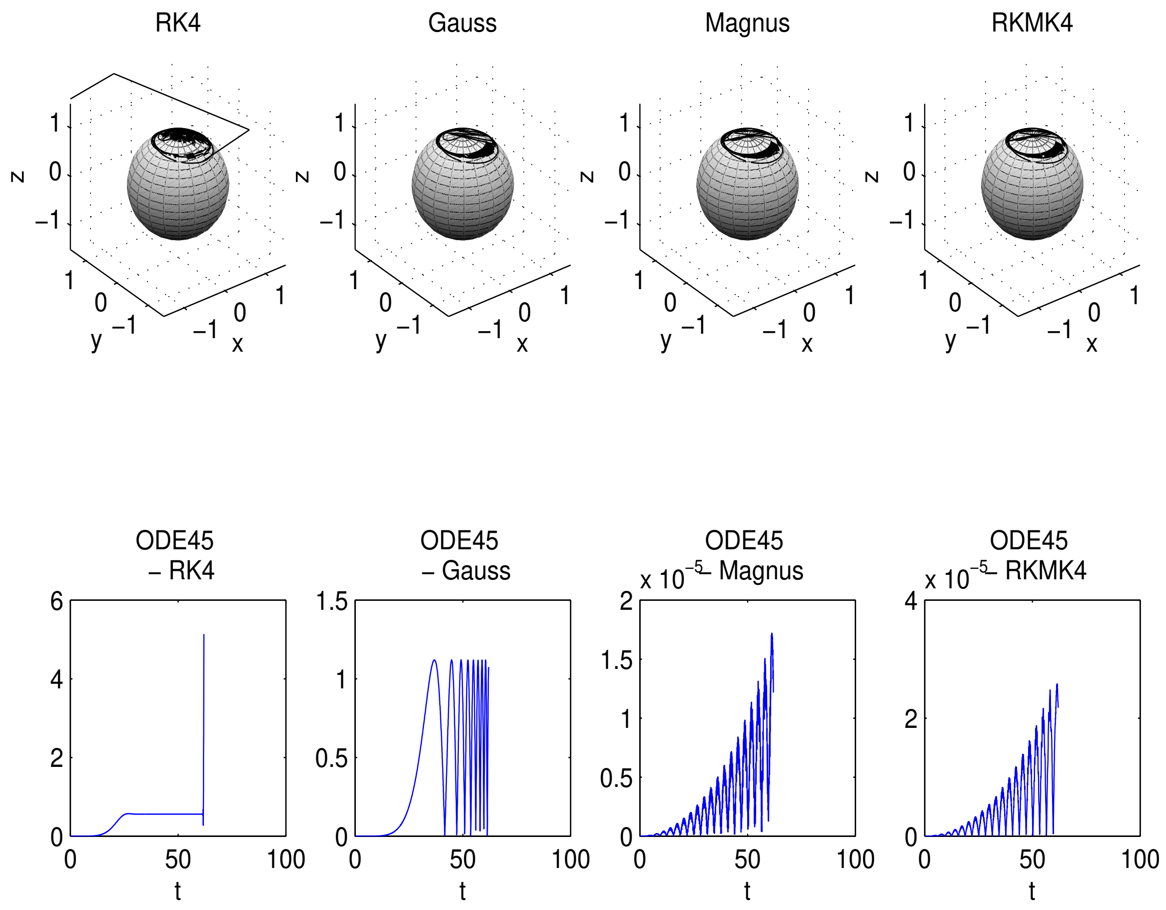


Figure 4.11: From solving Eq (4.10), the top 4 graphs are the solutions from each integrator plotted on the surface of a unit sphere. The bottom 4 graphs are the difference between solutions from ODE45 and each of the 4 integrators. $y_0 = [0, 0, 1]$, $t_0 = 0$, $t_f = 64$, and $h = 1/20$.

One of the advantage of using the Lie group methods such as the Magnus series expansions or the RKMK method, for solving this type of differential equations on the sphere is that the Lie group action Λ produces special orthogonal matrices, which, when multiplied to the vector $y(t) \in \mathbb{R}^3$, preserves its length in the rotation, so that solutions $y(t)$ will always remain on the sphere. This is not true if we use classical integrators such as RK method or the Gauss method. We can see this by observing the variation in the length of the solution vectors $y(t)$ with respect to the initial length, at each time step for all four integrators tested, in Figure 4.12.

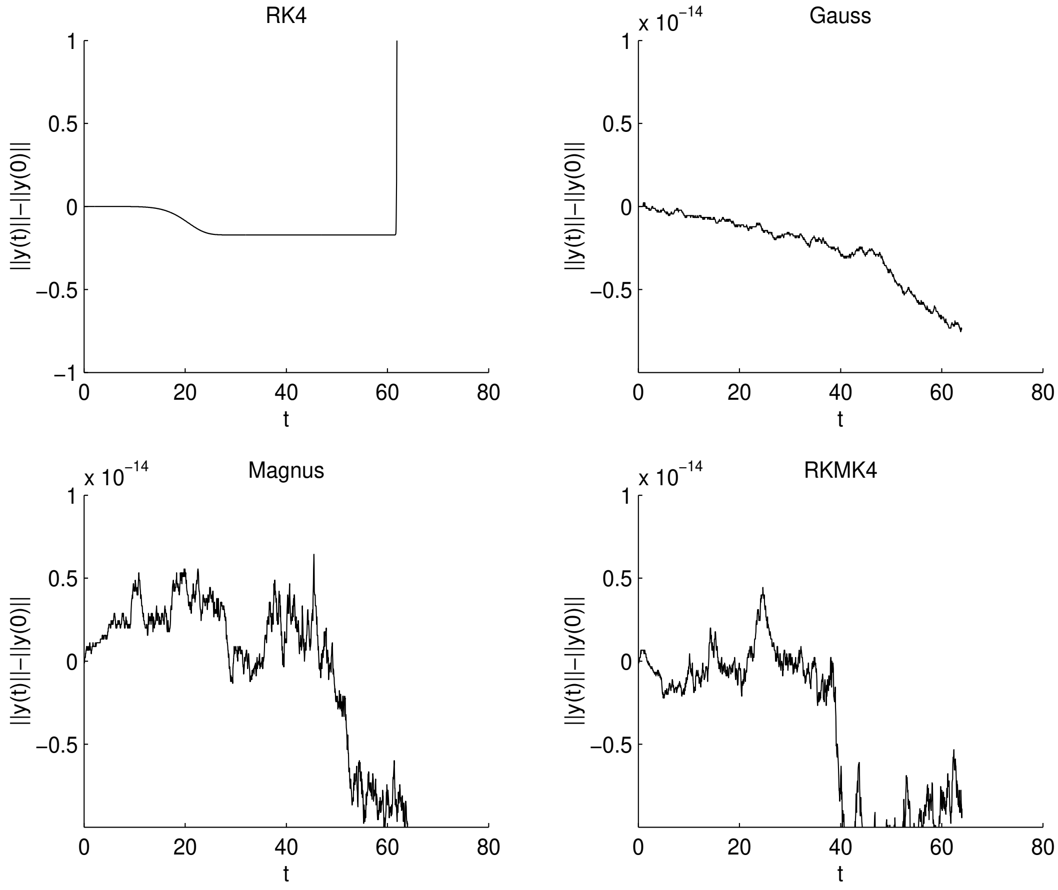


Figure 4.12: From solving Eq (4.10), the variation in the length of solution vectors against time t is plotted for each integrator. $y_0 = [0, 0, 1]$, $t_0 = 0$, $t_f = 64$, and $h = 1/20$.

4.5 Rigid Body Dynamics

In this section we will consider the dynamics of the rigid body in free motion (no external force), which can be represented in terms of the rotation matrix $A(t)$ in SO(3) about a fixed point (taken to be the centre of mass of the rigid body). This matrix $A(t) \in \mathrm{SO}(3)$ should act to preserve the distance of a point in the rigid body with respect to its centre of mass in rotation motion [33]. We will look closely at how the Euler Equations of motions for the rigid body can be transformed into a differential equation to be solved in the RKMK framework, and the important part that the coadjoint orbit plays in the abstract setting for the rigid body [8].

4.5.1 Systems of Particles

Before we can consider the motion of a system of particles, it is important to isolate this system from the rest of its environment. Then, the two main types of forces acting on the particles in the system are $F^{(e)}$, the 'external' forces due to the environment, and $F^{(i)}$, the 'internal' forces due to the interactions with other particles within the system. Hence, the force F_k acting on the k-th particle in the system is given by,

$$F_k = F_k^{(e)} + F_k^{(i)}.$$

Also, the system of particles should be defined in such a way that particles neither enter nor leave the system, so that conservation laws in classical mechanics can be applied.

The motion of a system of particles can be described by the motion of its centre of mass. This centre of mass for a system of point particles, each of mass m_k and radius r_k with respect to an origin, is defined to be,

$$\mathbf{R} = \frac{\sum\limits_k m_k r_k}{\sum\limits_k m_k}.$$

Furthermore, the motion of this entire system of particles can be thought of in terms of the motion of the centre of mass (translation), together with the motion about the centre of mass (rotation) given by the 'Principle of Moments'. If we consider the positions of the particles relative to the centre of mass $\mathbf{R}$, such that,

$$\widetilde{r}_k = r_k - \mathbf{R},$$

then, the 'principle of moments' is as stated below,

- The motion of the centre of mass,

$$\sum_k m_k \ddot{\mathbf{R}} = \sum_k F_k^{(e)}. \tag{4.11}$$

- The motion about the centre of mass,

$$\frac{d}{dt}\{\sum_k \widetilde{r}_k \times m_k \widetilde{v}_k\} = \sum_k \widetilde{r}_k \times F_k^{(e)}. \tag{4.12}$$

Note that $\ddot{\mathbf{R}}$ denotes the acceleration of the centre of mass. With respect to each particle within the system, $F_k^{(e)}$ is the acting external force, $p_k = m_k \widetilde{v}_k$ is the linear momentum, while $L_k = \widetilde{r}_k \times p_k$ and $N_k = \widetilde{r}_k \times F_k^{(e)}$ are the angular momentum and torque of each particle about the centre of mass. Hence, Eq (4.12) can be thought of as the rate of change of angular momentum of the entire system about its centre of mass, which is given by the sum of the torques due to the external forces acting on the system also calculated about the centre of mass. This relationship in rotational dynamics, $N = \frac{dL}{dt}$, is analogous to that of Newton's second law in translational motion, $F = \frac{dp}{dt}$.

4.5.2 Rotating Rigid Body

For a rotating object, we can either work in the space (inertial) frame, or the body frame. The space frame (denoted ^{s}), is fixed in space, from which a point on the object is observed to be moving, relative to this frame. Meanwhile, the body frame (denoted ^{b}), is a coordinate system which is rotating with the object, and so a point in the rigid body is stationary with respect to the body frame.

A rigid body is a system in which the distance between any two points within the system remains fixed during motion. If we consider only rotational dynamics of the rigid body, then there is a tensor relationship between its angular momentum L and angular velocity ω,

$$L = I\omega. \tag{4.13}$$

Unlike mass m in $p = mv$ for translational motion, the inertia tensor I has constant components only if the system is defined in the **body** coordinates. For a rigid body with a continuous distribution of mass, its density ρ is constant, and in the Cartesian coordinates, I is given by a symmetric matrix of volume integrals [33],

$$I = \begin{bmatrix} \int(y^2+z^2)\rho dV & -\int xy\rho dV & -\int xz\rho dV \\ -\int yx\rho dV & \int(z^2+x^2)\rho dV & -\int yz\rho dV \\ -\int zx\rho dV & -\int zy\rho dV & \int(x^2+y^2)\rho dV \end{bmatrix}. \tag{4.14}$$

From linear algebra, given that the inertia tensor I is symmetric, then it is orthogonally diagonalisable. The three mutually orthogonal eigenvectors of the diagonalised I are the **principal axes** of the rigid body, and the eigenvalues I_1, I_2, and I_3 are the **principal moments of inertia**. We can use the principal axes as the body frame of the system because the principal axes are fixed in the body. This means that as the rigid body rotates about a fixed point, so do the directions of the principal axes. Furthermore, now that I is diagonalised, the relationship in Eq (4.13) is simplified to $L_x = I_1\omega_x$, $L_y = I_2\omega_y$, and $L_z = I_3\omega_z$, which is true only in this coordinate system. In the space frame however, generally the quantities in Eq (4.13) will depend on time t as the object rotates in space.

To simplify calculations, from now on, we will consider the rotational dynamics of the rigid body about its centre of mass. Also, we will take the body coordinates to be the principal axes of the rigid body.

For a rotating system of particles, let us assume that at $t = 0$, the space and body coordinates coincide. This means that the position of a particle with respect to the centre of mass at time $t = 0$ is given by $r(0)$, then $r^s(0) = r^b(0)$. In a rigid body rotation, the new position of this particle in the space coordinate at time t, depends on a 3×3 rotation matrix $A(t)$, such that,

$$r^s(t) = A(t)r^b(0).$$

Note that in the body frame, the position vector $r^b \in \mathbb{R}^3$ is constant, because the position of the particle on the rigid body will not be moving with respect to the body frame. Hence we can rewrite the above equation as,

$$r^s(t) = A(t)r^b. \tag{4.15}$$

The rotation matrix $A(t)$ in Eq (4.15) is in the Special Orthogonal Lie group SO(3). $A(t)$ is an orthogonal matrix such that $A^TA = \mathrm{Id}$ (the identity matrix), since a rotation of any particles in the rigid body must preserve length and angles, while $\det(A) = 1$ so that the reflections will give the correct handedness of a triads of vectors. Furthermore, $A(t)$ can be found by solving a system of ordinary differential equations, as explained below.

For a particle in the rotating rigid body, its velocity in Cartesian coordinates in the space frame is given by,

$$v^s(t) = \omega^s(t) \times r^s(t) = \begin{bmatrix} 0 & -\omega_3^s & \omega_2^s \\ \omega_3^s & 0 & -\omega_1^s \\ -\omega_2^s & \omega_1^s & 0 \end{bmatrix} r^s(t) = \overline{\omega^s(t)}r^s(t). \tag{4.16}$$

where $\overline{\omega^s}$ denotes the skew-symmetric $\mathfrak{so}(3)$ matrix representation of the angular velocity ω^s vector in $\mathbb{R}^3$.

Note that in this case, if we consider the basis of $\mathfrak{so}(3)$, given by the set of matrices e_1, e_2, e_3,

$$e_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}, \quad e_2 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix}, \quad e_3 = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

then, the matrix representation $\overline{\omega}$ is a linear combination of the following,

$$\overline{\omega} = \omega_1 e_1 + \omega_2 e_2 + \omega_3 e_3.$$

This means that the vector representation $\omega = [\omega_1, \omega_2, \omega_3]^T$ simply contains the coefficients to the above linear combination for the matrix $\overline{\omega} \in \mathfrak{so}(3)$.

To derive the differential equation for the rotation matrix $A(t)$, we can differentiate Eq (4.15) with respect to t, and substitute the same equation into this velocity expression,

$$v^s(t) = \dot{A}(t)r^b = \dot{A}(t)(A^{-1}(t)r^s(t)).$$

Comparing the above with Eq (4.16) yields the following,

$$\overline{\omega^s(t)} = \dot{A}(t)A^{-1}(t), \tag{4.17}$$

which can be rearranged into the differential equation below, with I being the 3×3 identity matrix. (This should not be confused with the notation for the inertia tensor I in Eq (4.13).)

$$\dot{A}(t) = \overline{\omega^s(t)}A(t), \quad A(0) = \mathrm{Id}. \tag{4.18}$$

This means that if ω^s of the rotating rigid body in the space frame is known for a particular time t, then we can determine the rotation matrix $A(t)$ by solving the initial value problem in Eq (4.18). It is also interesting to note that the angular velocity $\omega^b(t)$ in the body coordinates obeys the following relationship in its matrix representation,

$$\overline{\omega^b(t)} = A^{-1}(t)\dot{A}(t). \tag{4.19}$$

If the angular velocity of the rigid body $\omega^b(t)$, in the body frame is known, then the rotation matrix $A(t)$ can be found by solving a differential equation similar to Eq (4.18).

$$\dot{A}(t) = A(t)\overline{\omega^b(t)}, \quad A(0) = \mathrm{Id}. \tag{4.20}$$

The proof to Eq (4.19) and Eq (4.20) can be found in either [8] or [33].

With the two differential equations Eq (4.18) and Eq (4.20) for the rotation matrix $A(t)$, we can equate and rearrange the two equations to find the following relationship between the matrix representations of ω^s and ω^b,

$$\overline{\omega^s(t)} \quad = \quad A(t)\overline{\omega^b(t)}A(t)^{-1}.$$

The above is simply the Adjoint representation of the Lie group in Definition 1.13. Hence, for the rigid body, the Adjoint representation is,

$$\overline{\omega^s(t)} = \mathrm{Ad}_{A(t)}(\overline{\omega^b(t)}) = A(t)\overline{\omega^b(t)}A(t)^{-1}. \tag{4.21}$$

Furthermore, in terms of the skew-symmetric matrices in $\mathfrak{so}(3)$, it can be proved that the above Adjoint matrix representation is equivalent to the following vector representation,

$$\mathrm{Ad}_A(\overline{\omega^b}) \twoheadrightarrow A\omega^b, \tag{4.22}$$

where ω^b without $\overline{}$ is the angular velocity in the body coordinates, in the vector notation $\mathbb{R}^3$. The notation $\twoheadrightarrow$ denotes a transformation from the skew-symmetric matrix on the left hand side to its 3-component vector form on the right hand side, as explained earlier in this section using basis in $\mathfrak{so}(3)$. Similarly, $\twoheadleftarrow$ denotes a change of representation from vector on the left hand side to skew-symmetric matrix on the right hand side.

Proof. There are several ways of proving Eq (4.22). One way is to make use of the following relationships for matrices $\overline{v}, \overline{w} \in \mathfrak{so}(3)$,

$$\begin{aligned}
\mathrm{ad}^l_{\overline{v}}(\overline{w}) &= (-1)^l \sum_{j=0}^{l} (-1)^j \begin{pmatrix} l \\ j \end{pmatrix} \overline{v}^j\, \overline{w}\, \overline{v}^{l-j}, && (4.23)\\
\mathrm{ad}^k_{\overline{v}}(\overline{w}) &\twoheadrightarrow \overline{v}^k \cdot w, && (4.24)\\
\mathrm{Ad}_{\exp(\overline{v})}(\overline{w}) &= \exp(\mathrm{ad}_{\overline{v}})(\overline{w}). && (4.25)
\end{aligned}$$

Varadarajan in [34] proved by induction that Eq (4.23) is true for the adjoint operator $\mathrm{ad} : \mathfrak{g} \times \mathfrak{g} \to \mathfrak{g}$. For details, please refer to Chapter 2 of [34]. The relation in Eq (4.24) can also be proved by induction. Let $v = [v_1, v_2, v_3]$ and $w = [w_1, w_2, w_3]$ be vectors in $\mathbb{R}^3$. For their skew-symmetric matrix representations $\overline{v}$ and $\overline{w}$ in $\mathfrak{so}(3)$, we have,

$$\overline{v} = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix} \quad \text{and} \quad \overline{w} = \begin{bmatrix} 0 & -w_3 & w_2 \\ w_3 & 0 & -w_1 \\ -w_2 & w_1 & 0 \end{bmatrix}.$$

To see that Eq (4.24) is true for $k = 1$, we do a straight forward calculation of $\mathrm{ad}_{\overline{v}}(\overline{w})$.

$$\begin{aligned}
\mathrm{ad}_{\overline{v}}(\overline{w}) &= [\overline{v}, \overline{w}] = \begin{bmatrix} 0 & v_2w_1 - w_2v_1 & v_3w_1 - w_3v_1 \\ v_1w_2 - w_1v_2 & 0 & v_3w_2 - w_3v_2 \\ v_1w_3 - w_1v_3 & v_2w_3 - w_2v_3 & 0 \end{bmatrix} \\
&\twoheadrightarrow \begin{pmatrix} v_2w_3 - w_2v_3 \\ v_3w_1 - w_3v_1 \\ v_1w_2 - w_1v_2 \end{pmatrix} = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix} \begin{bmatrix} w_1 w_2 w_3 \end{bmatrix} = \overline{v} \cdot w.
\end{aligned}$$

Assuming that Eq (4.24) is true for $k > 1$, we have,

$$\mathrm{ad}^k_{\overline{v}}(\overline{w}) = \overline{\zeta} \twoheadrightarrow \overline{v}^k \cdot w = \zeta.$$

Note that since multiplying skew-symmetric matrices together yields another skew-symmetric matrix, then $\overline{\zeta}$ is a skew-symmetric matrix in $\mathfrak{so}(3)$, and ζ is the vector representation in $\mathbb{R}^3$. By induction, Eq (4.24) for $k+1$, has the expression,

$$\mathrm{ad}^{k+1}_{\overline{v}}(\overline{w}) = \mathrm{ad}_{\overline{v}}(\mathrm{ad}^k_{\overline{v}}(\overline{w})) = \mathrm{ad}_{\overline{v}}(\overline{\zeta}) = \left[\overline{v}, \overline{\zeta}\right] \twoheadrightarrow \overline{v} \cdot \zeta = \overline{v}(\overline{v} \cdot w) = \overline{v}^{k+1} \cdot w.$$

Hence, Eq (4.24) is true for all $k > 0$.

For Eq (4.25). we can prove it by working with both sides of the equation, using the matrix exponential. From the left hand side we have,

$$\mathrm{Ad}_{\exp(\overline{v})}(\overline{w}) = \exp(\overline{v})\overline{w}\exp(-\overline{v}) = \sum_{i=0}^{\infty}\sum_{k=0}^{\infty}(-1)^k\frac{\overline{v}^i\,\overline{w}\,\overline{v}^k}{i!k!}.$$

From the right hand side of Eq (4.25), we substitute Eq (4.23) and rearrange the summations,

$$\begin{aligned}
\exp(\mathrm{ad}_{\overline{v}})(\overline{w}) &= \sum_{l=0}^{\infty}\frac{\mathrm{ad}^l_{\overline{v}}(\overline{w})}{l!} = \sum_{l=0}^{\infty}\frac{1}{l!}(-1)^l\sum_{j=0}^{l}(-1)^j\begin{pmatrix} l \\ j \end{pmatrix}\overline{v}^j\,\overline{w}\,\overline{v}^{l-j} \\
&= \sum_{l=0}^{\infty}\sum_{j=0}^{l}(-1)^{l+j}\frac{\overline{v}^j\,\overline{w}\,\overline{v}^{l-j}}{(l-j)!j!} = \sum_{l=0}^{\infty}\sum_{j=0}^{l}(-1)^{l-j}\frac{\overline{v}^j\,\overline{w}\,\overline{v}^{l-j}}{(l-j)!j!} \\
&= \sum_{j=0}^{\infty}\sum_{l=j}^{l}(-1)^{l-j}\frac{\overline{v}^j\,\overline{w}\,\overline{v}^{l-j}}{(l-j)!j!} = \sum_{i=0}^{\infty}\sum_{k=0}^{l}(-1)^k\frac{\overline{v}^i\,\overline{w}\,\overline{v}^k}{(k)!i!} \qquad \text{let } j = i \text{ and } k = l - j,
\end{aligned}$$

which is the left hand side of Eq (4.25), and hence the relation is true.

Finally, to prove Eq (4.22), we use the matrix exponential in Eq (4.25) and substitute in the matrix-to-vector relationship in Eq (4.24), such that,

$$\begin{aligned}
\mathrm{Ad}_{\exp(\overline{v})}(\overline{w}) &= \exp(\mathrm{ad}_{\overline{v}})(\overline{w}) = \overline{w} + \mathrm{ad}_{\overline{v}}(\overline{w}) + \frac{1}{2!}\mathrm{ad}^2_{\overline{v}}(\overline{w}) + \cdots \\
&\twoheadrightarrow w + \overline{v} \cdot w + \frac{1}{2!}\overline{v}^2 \cdot w + \cdots = (\mathrm{Id} + \overline{v} + \frac{1}{2!}\overline{v}^2 + \cdots) \cdot w = \exp(\overline{v}) \cdot w.
\end{aligned}$$

Given that $A \in \mathrm{SO}(3)$, we can always find its corresponding Lie algebra element $\overline{v} \in \mathfrak{so}(3)$, or in terms of a set of basis in $\mathfrak{so}(3)$. Hence, A can be written as $\exp(\overline{v})$, and in which case, the above representation is in the form of Eq (4.22). □

4.5.3 Euler Equations of Motion

The Euler Equations of Motion can be derived by considering the expressions for the torque N^b of the system in the body frame, where its 3 Cartesian coordinates are given by,

$$\begin{bmatrix} N^b_1 \\ N^b_2 \\ N^b_3 \end{bmatrix} = \begin{bmatrix} (\frac{dL^b}{dt} + \omega^b \times L^b)_1 \\ (\frac{dL^b}{dt} + \omega^b \times L^b)_2 \\ (\frac{dL^b}{dt} + \omega^b \times L^b)_3 \end{bmatrix}$$

From Section 4.5.2, if the body coordinates are aligned with the principal axes of the rigid body, then the inertia tensor is a constant positive diagonal matrix of principal moments of inertias I_1, I_2, and I_3. Then, the angular momentum components in the body coordinates are $L_1^b = I_1\omega_1^b$, $L_2^b = I_2\omega_2^b$, and $L_3^b = I_3\omega_3^b$. Substituting such components into the above equation for N^b, we derive a system of three differential equations called the Euler Equations of Motion in the body coordinates [33].

$$\begin{aligned}
I_1\frac{d\omega_1^b}{dt} &= (I_2 - I_3)\omega_2^b\omega_3^b + N_3^b \\
I_2\frac{d\omega_2^b}{dt} &= (I_3 - I_1)\omega_3^b\omega_1^b + N_2^b \\
I_3\frac{d\omega_3^b}{dt} &= (I_1 - I_2)\omega_1^b\omega_2^b + N_z^b.
\end{aligned} \tag{4.26}$$

This system of equations allow us to find the value of angular velocity $\omega^b(t)$ in the body coordinates at time t, which is what we need to solve for the rotation matrix $A(t)$ in Eq (4.20) for the orientation of the rotating rigid body. In fact, solving both sets of differential equations in Eq (4.20) for $A(t)$ and Eq (4.26) for $\omega^b(t)$ will completely determine the position $r^s(t)$ of the rotating rigid body in the space frame by an update with Eq (4.15).

Free rotation of a rigid body is when there is no torque acting on the system, and $N = 0$. For rotational motion, we have $N = \frac{dL^s}{dt}$, which implies that the angular momentum L^s in the space coordinates is conserved. So, L^s is fixed both in magnitude and direction,

$$N = \frac{dL^s}{dt} = 0, \qquad \Rightarrow L^s = \text{constant}.$$

Note that although L^s is fixed in space, if we consider its components with respect to the body coordinates L^b, it is not constant. With a little rearrangement, we can write the above Euler Equation for free motion in the following form, (recall that in the body frame, $L_i^b = I_i\omega_i^b$.),

$$\dot{L}^b = \begin{pmatrix} \dot{L}_1^b \\ \dot{L}_2^b \\ \dot{L}_3^b \end{pmatrix} = -\begin{pmatrix} 0 & -\omega_3^b & \omega_2^b \\ \omega_3^b & 0 & -\omega_1^b \\ -\omega_2^b & \omega_1^b & 0 \end{pmatrix}\begin{pmatrix} L_1^b \\ L_2^b \\ L_3^b \end{pmatrix} = -\overline{\omega^b}L^b. \tag{4.27}$$

Once again, the $\overline{}$ denotes the skew-symmetric matrix representation of the angular velocity vector $\omega^b \in \mathbb{R}^3$. By solving Eq (4.27) along with Eq (4.20) with a given initial $\omega^b(t)$, this yields the orientation $r^s(t)$ of the rotating rigid body in free motion. Indeed, when using the classical integrators such as Runge-Kutta method or the implicit Gauss method in the following section, the two differential equations to be solved are Eq (4.27) and Eq (4.20).

One important point to note is that even for free rotations where $N = 0$, the motion of the rigid body can be quite complicated. The angular velocity ω^b is not constant in general. In fact, ω^b is constant in free rotation only if the rigid body is rotating precisely about one of the principal axes, so that the initial angular velocity $\omega^b(0)$ is aligned with one of the axes. Even slight deviation of $\omega^b(0)$ from the principal axis results in oscillations of $\omega^b(t)$ in the body frame, and the body appears to 'wobble' in rotation.

Furthermore, for a rigid body with distinct moments of inertia, say $I_1 > I_2 > I_3$, its motion in free rotation can become unstable. If the object initially rotates with a large component of the initial angular velocity $\omega^b(0)$ in the direction of the principal axis associated with I_2, and with small components in the other two directions, then it is found that $\omega^b(t)$ calculated from the Euler Equations rapidly deviates from the principal axis associated with I_2. This results in what's known as the **unstable rotation** of the rigid body [33]. For the analytical details of the free motion of the rigid body, please refer to Chapter 15 of [24].

4.5.4 Energy and Coadjoint Orbits of the Rigid Body

Suppose that the rigid body problem we want to consider is in free rotation ($N = 0$), then the Hamiltonian energy $H(t)$ of this system is given by its kinetic energy,

$$H(t) = \frac{1}{2}\omega(t)^T \cdot I \cdot \omega(t), \tag{4.28}$$

where I is the 3×3 inertia tensor, $\omega(t)$ is the angular velocity of the rigid body in vector form $\mathbb{R}^3$, and ^{T} denotes transpose of the vector. From the conservation of energy, we know that $H(t)$ is invariant, and in this case, it is also a quadratic function.

In a more general notation, the energy of the rigid body can be considered as an inner product $\langle \cdot, \cdot \rangle$ on the Lie algebra $\mathfrak{so}(3)$, and one can define the angular momentum L^b in the body coordinates as an element of the dual space $\mathfrak{so}(3)^*$ [8]. A dual space of a vector space V is the set of linear functions, $f : V \to \mathbb{R}$, and is denoted V^*. To understand this when applied to Lie algebra $\mathfrak{g}$, we need the following definitions [31].

Definition 4.18. *Let $(e_1, \ldots, e_n)$ be the ordered basis for a Lie algebra $\mathfrak{g}$ of dimension n, and let $(\epsilon_1, \ldots, \epsilon_n)$ be the dual basis for the dual space $\mathfrak{g}^*$, such that,*

$$\epsilon_i(e_j) = \delta_{ij}, \quad 1 \leq i, j \leq n,$$

where δ_{ij} is the Kronecker-delta function. For any linear operator $D : \mathfrak{g} \to \mathfrak{g}$, we denote by $D^* : \mathfrak{g}^* \to \mathfrak{g}^*$ the dual operator, such that for any $\phi \in \mathfrak{g}^*$ and $v \in \mathfrak{g}$,

$$(D^*\phi)(v) = \phi(Dv).$$

In terms of the rigid body, the angular momentum L can be written as a linear combination of the dual basis such that, $L = L_1\epsilon_1 + L_2\epsilon_2 + L_3\epsilon_3$. So, with respect to the basis in $\mathfrak{so}(3)$ and its dual basis in $\mathfrak{so}(3)^*$, we have L^b as an element in the dual space $\mathfrak{so}(3)^*$ in the following sense,

$$L^b = \sum_i I_{ii}\epsilon_i(\sum_j \omega_j^b e_j)\epsilon_i = \sum_i I_{ii}\omega_i^b \epsilon_i = I \cdot \omega^b,$$

where L^b is a vector in $\mathbb{R}^3$. The above is simply Eq (4.13) in the body coordinates. In this abstract setting, the inertia tensor I is considered as a mapping from $\mathfrak{g} \to \mathfrak{g}^*$. In our case for the rigid body, where I is symmetric and positive definite, then the inner product on $\mathfrak{g}$ is defined as,

$$\langle u, v \rangle = (Iu)(v) = (Iv)(u).$$

Hence, along with the definitions for the bases for $\mathfrak{so}(3)$ and $\mathfrak{so}(3)^*$, we can consider the energy E in Eq (4.28) as an inner product on $\mathfrak{so}(3)$ in the following way [8],

$$\begin{aligned} E &= \frac{1}{2}(\omega^b)^T I \omega^b = \frac{1}{2}\sum_i I_{ii}\epsilon_i(\sum_j \omega_j^b e_j)\epsilon_i(\sum_k \omega_k^b e_k)\epsilon_i \\ &\twoheadleftarrow \frac{1}{2}\langle \overline{\omega^b}, \overline{\omega^b} \rangle = \frac{1}{2}(I\overline{\omega^b})(\overline{\omega^b}) = \frac{1}{2}L^b(\overline{\omega^b}). \end{aligned} \tag{4.29}$$

We now have two representations for the kinetic energy E, as stated in the following,

$$E = \frac{1}{2}(L^b)^T \omega^b \twoheadleftarrow \frac{1}{2}L^b(\overline{\omega^b}). \tag{4.30}$$

Note that the energy is independent of the coordinate system, then together with the Adjoint representation in Eq (4.21), we can derive yet another equivalent expression for the kinetic energy [8],

$$E \twoheadleftarrow \frac{1}{2}L^b(\overline{\omega^b}) = \frac{1}{2}L^s(\overline{\omega^s}) = \frac{1}{2}L^s(\mathrm{Ad}_A(\overline{\omega^b})) = \frac{1}{2}(\mathrm{Ad}_A^* L^s)(\overline{\omega^b}), \tag{4.31}$$

where matrix $A \in \mathrm{SO}(3)$. Recall that in Definition 4.18, the dual operator D^* is defined. In the case of the rigid body, where $L^s \in \mathfrak{so}(3)^*$, $\overline{\omega^b} \in \mathfrak{so}(3)$, the dual of the Adjoint operator Ad can be used, as stated in Eq (4.31). Comparing Eq (4.29) and Eq (4.31), we find,

$$L^b = \mathrm{Ad}_A^* L^s.$$

Furthermore, we can find out what Ad_A^* is, if we consider the vector-to-matrix ($\twoheadleftarrow$) and matrix-to-vector ($\twoheadrightarrow$) representations, as similar in Eq (4.30),

$$E \twoheadleftarrow \frac{1}{2}L^s(\mathrm{Ad}_A(\overline{\omega^b})) \twoheadrightarrow \frac{1}{2}(L^s)^T(A \cdot \omega^b) = \frac{1}{2}(A^T L^s)^T \omega^b.$$

Hence, comparing the above with Eq (4.30), this yields the relationship which we will make use of, when solving the rigid body problem using the RKMK method.

$$L^b(t) = A(t)^T L^s = A(t)^{-1} L^s, \tag{4.32}$$

where $A(t) \in \mathrm{SO}(3)$, so $A(t)^T = A(t)^{-1}$. Also, L^s is a vector of constants in $\mathbb{R}^3$ for free rotational motion, and $L^b(t)$ changes according to the rotation matrix $A(t)$ at any given time t.

Equation (4.32) also defines the coadjoint orbit $\mathfrak{O}_{L^s}^* \subset \mathfrak{so}(3)^*$, which is defined in the following way,

Definition 4.19. *For any $m \in \mathfrak{g}^*$, we can define the coadjoint orbits $\mathfrak{O}_m^* \subset \mathfrak{g}^*$ as,*

$$\mathfrak{O}_m^* = \{p \in \mathfrak{g}^* : p = \mathrm{Ad}_g^*(m),\ g \in G.\}$$

Since $N = \frac{dL^s}{dt} = 0$ for free rotation, and

$$L^b = \mathrm{Ad}_A^* L^s = A^T L^s,$$

then we can see that the angular momentum in the body coordinates $L^b(t)$ remains in the same coadjoint orbit $\mathfrak{O}_{L^s}^*$ for all time.

4.5.5 Rotation of a Rectangular Block

Let us consider a rectangular block of uniformly distributed mass as a rigid body rotating in free motion. Define the origin of this rectangular block as its centre of mass, and let the principal Cartesian axes act as its body frame. If we define the block as having the following dimensions,

$$|x| \le \frac{1}{2}a, \quad |y| \le \frac{1}{2}b, \quad |z| \le \frac{1}{2}c,$$

where $a = 2$, $b = 4$, $c = 6$, mass $m = \rho abc = 12$. By Eq (4.14) in Section 4.5.2, its inertia tensor is given by a diagonal matrix of constants,

$$I = \begin{bmatrix} I_1 & 0 & 0 \\ 0 & I_2 & 0 \\ 0 & 0 & I_3 \end{bmatrix} = \begin{bmatrix} \frac{m}{12}(b^2+c^2) & 0 & 0 \\ 0 & \frac{m}{12}(c^2+a^2) & 0 \\ 0 & 0 & \frac{m}{12}(a^2+b^2) \end{bmatrix} = \begin{bmatrix} 52 & 0 & 0 \\ 0 & 40 & 0 \\ 0 & 0 & 20 \end{bmatrix}. \tag{4.33}$$

Since each of the principal moments of inertia are distinct, then from the discussion in Section 4.5.3, if the rectangular block is rotating in free motion, then a good geometric integrator should produce numerical results for which the angular momentum $L^s = I(t)\omega^s(t)$ of the block is constant in both direction and magnitude in the space frame. Because the body principal axes are rotating with the rectangular block, then $L^b(t)$ in the body frame will seem to rotate with respect to the body axes, but its magnitude $\|L^b(t)\|$, should still be constant with respect to time t. Also, the Hamiltonian $H(t)$, which in this case, is given by the kinetic energy of the rigid body in Eq (4.28), is a quadratic function. It would be interesting to perform some numerical experiments using the RKMK method discussed in Chapter 3, and comparing the results with those of classical RK method. Furthermore, it is well known that symplectic method such as the implicit Gauss method preserves quadratic invariant. It will also be of interest to see how well the performance of RKMK method compares with such symplectic integrators.

4.5.6 Rigid Body Problem in the RKMK Framework

Before beginning to integrating the ODE problem associated with the rotation of the rectangular block given in Section 4.5.5, one must understand how the rigid body problem fits into the general RKMK framework. For example, we need to find the function $f : G \to \mathfrak{g}$ and choose a coordinate map which allows us to transform the existing differential equation on $\mathcal{M}$ into one which evolves on $\mathfrak{g}$. Several authors had done this [14], [19], and [8], especially Lewis and Simo, who pioneered the designing of the Lie group methods for solving the low order rigid body problems [8]. But in this present discussion, we will follow the detailed work of Owren and Celledoni in [8], which emphasises the coadjoint action in the rigid body equation and derived the function $f : G \to \mathfrak{g}$ by the Adjoint representation of the matrices in $\mathfrak{so}(3)$.

For the rigid body problem, the manifold $\mathcal{M} = \mathfrak{so}(3)^*$, where the angular momentum $L^b(t)$ in the body coordinates is a curve in this dual space. The acting Lie group is the special orthogonal group $G = \mathrm{SO}(3)$, and the Lie algebra is the group of skew-symmetric matrices in $\mathfrak{so}(3)$. Thus, locally around a point $L \in \mathcal{M} = \mathfrak{so}(3)^*$, the solution curve $L^b(t) \in \mathfrak{so}(3)^*$ is transformed to a curve $u(t) \in \mathfrak{so}(3)$ through 0, via the curve through the identity element in $G = \mathrm{SO}(3)$ by the Lie group action [8]. To find this function $f : \mathrm{SO}(3) \to \mathfrak{so}(3)$, we follow the steps below [8].

If we differentiate Eq (4.32) with respect to t, we have,

$$\frac{dL^b(t)}{dt} = -A(t)^{-1}\frac{dA(t)}{dt}A(t)^{-1}L^s. \tag{4.34}$$

Proof. This can be done either by first principles, where,

$$\begin{aligned}
\frac{dL^b(t)}{dt} &= \lim_{h\to 0}\frac{L^b(t+h)-L^b(t)}{h} = \lim_{h\to 0}\frac{A(t+h)^{-1}L^s - A(t)^{-1}L^s}{h} \\
&= \lim_{h\to 0}\frac{\left[A(t)+hA'(t)+O(h^2)\right]^{-1}L^s - A(t)^{-1}L^s}{h} \\
&= \lim_{h\to 0}\frac{\left[A(t)(I+hA(t)^{-1}A'(t)+O(h^2))\right]^{-1}L^s - A(t)^{-1}L^s}{h} \\
&= \lim_{h\to 0}\frac{\left[\frac{I-hA(t)^{-1}A'(t)+O(h^2)}{I+O(h^2)}\right]A(t)^{-1}L^s - A(t)^{-1}L^s}{h} \\
&= \left[A(t)^{-1}L^s - A(t)^{-1}A'(t)A(t)^{-1}L^s\right] - A(t)^{-1}L^s \\
&= -A(t)^{-1}A'(t)A(t)^{-1}L^s.
\end{aligned}$$

Alternatively, Prof. Owren suggested the following proof based on the product rule where, for 2 matrices $u(t)$ and $v(t)$ so that $u(t) \cdot v(t)$ is well defined,

$$\frac{d}{dt}(u(t) \cdot v(t)) = \dot{u}(t)v(t) + u(t)\dot{v}(t).$$

Then, starting with $L^s = A(t)L^b$, differentiate with respect to t,

$$\begin{aligned} \frac{d}{dt}L^s &= \frac{d}{dt}A(t)L^b(t) \\ 0 &= \dot{A}(t)L^b(t) + A(t)\dot{L}^b(t) \\ \Rightarrow \dot{L}^b(t) &= -A(t)^{-1}\dot{A}(t)L^b(t) \\ &= -A(t)^{-1}\dot{A}(t)A(t)^{-1}L^s. \end{aligned}$$

□

Next, we want to make the following change of variables such that in a local region near the point $p \in G = \mathrm{SO}(3)$, the rotation matrix $A(t)$ is given by the right trivialisation of the Lie group action, defined in Definition 1.12,

$$A(t) = \Psi(u(t)) \cdot p, \quad u(t) \in \mathfrak{so}(3). \tag{4.35}$$

The coordinate mapping $\Psi : \mathfrak{g} \to G$ can be either the exp map or the Cayley map. Note that the left trivialisation of this Lie group action can also be used as the change of variables. Substituting Eq (4.35) into Eq (4.34) yields the following,

$$\begin{aligned} \dot{L}^b(t) &= -A(t)^{-1}\dot{A}(t)A(t)^{-1}L^s \\ &= -(\Psi(u(t)) \cdot p)^{-1}\frac{d}{dt}(\Psi(u(t)) \cdot p)(\Psi(u(t)) \cdot p)^{-1}L^s \\ &= -(p^{-1}\Psi(u(t))^{-1})d\Psi_{u(t)}(\dot{u}(t))\Psi(u(t))p(p^{-1}\Psi(u(t))^{-1})L^s \\ &= -(p^{-1}\Psi(u(t))^{-1})d\Psi_{u(t)}(\dot{u}(t))L^s. \end{aligned} \tag{4.36}$$

In the above, we have used the fact that $G = \mathrm{SO}(3)$, so its inverse is the transpose, and the relationship between the time derivative of Ψ and its differential $d\Psi$,

$$\frac{d}{dt}\Psi(u(t)) = d\Psi_{u(t)}(\dot{u}(t))\Psi(u(t)).$$

Note that if $\Psi = \exp$, then the above relation is given by Theorem 1.3.

Now recall that from the Euler Equation for free motion, we have,

$$\dot{L}^b(t) = -\overline{\omega^b(t)}L^b(t), \tag{4.37}$$

where $\overline{\omega^b}$ is the skew-symmetric matrix representation for $\omega^b \in \mathbb{R}^3$. If we substitute Eq (4.32) and Eq (4.35) into Eq (4.37) we have,

$$\dot{L}^b(t) = -\overline{\omega^b(t)}A(t)^{-1}L^s = -\overline{\omega^b(t)}(p^{-1}\Psi(u(t))^{-1})L^s. \tag{4.38}$$

Now, equate Eq (4.36) and Eq (4.38) where the two time-dependent matrix equations whose actions on the constant vector $L^s \in \mathbb{R}^3$ are the same, we have,

$$\begin{aligned} -p^{-1}\Psi(u(t))^{-1}d\Psi_{u(t)}(\dot{u}(t))L^s &= -\overline{\omega^b(t)}p^{-1}\Psi(u(t))^{-1}L^s \\ d\Psi_{u(t)}(\dot{u}(t)) &= \Big(\Psi(u(t))p\Big)\overline{\omega^b(t)}\Big(\Psi(u(t))p\Big)^{-1} = \mathrm{Ad}_{\Psi(u(t))p}\left(\overline{\omega^b(t)}\right) \\ &\twoheadrightarrow \Psi(u(t))p \cdot \omega^b(t), \end{aligned}$$

where $\twoheadrightarrow$ denotes the change from matrix-to-vector representation of the Adjoint operator Ad given in Eq (4.22).

We can also write $\omega^b(t)$ in terms of the inertia tensor I and the constant L^s, by Eq (4.13), Eq (4.32), and Lie group action in Eq (4.35),

$$\omega^b(t) = I^{-1}L^b(t) = I^{-1}A(t)^{-1}L^s = I^{-1}p^{-1}\Psi(u(t))^{-1}L^s.$$

So, we can now rewrite the vector representation of the differential $d\Psi_{u(t)}(\dot{u}(t))$ based on $u(t) \in \mathfrak{so}(3)$ and other known quantities such as L^s and I. From here, we can rearrange the equation to yield a differential equation for $u(t)$ with respect to t,

$$\begin{aligned} d\Psi_{u(t)}(\dot{u}(t)) &\twoheadrightarrow \Psi(u(t))pI^{-1}p^{-1}\Psi(u(t))^{-1}L^s \twoheadleftarrow \overline{\Psi(u(t))pI^{-1}p^{-1}\Psi(u(t))^{-1}L^s} \\ \Rightarrow \dot{u}(t) &= d\Psi^{-1}_{u(t)}\left(\overline{\Psi(u(t))pI^{-1}p^{-1}\Psi(u(t))^{-1}L^s}\right). \end{aligned} \tag{4.39}$$

In other papers such as [28], this differential equation $\dot{u}(t)$ evolving on the linear vector space $\mathfrak{g}$ is written as,

$$\dot{u}(t) = d\Psi^{-1}\left(f(\lambda_p(u))\right).$$

In this case for the rigid body, the Lie algebra action $\lambda_p(u)$ is the change of variables in Eq (4.35),

$$\lambda_p(u(t)) = \Lambda(\Psi(u(t)), p) = \Psi(u(t)) \cdot p = A(t), \tag{4.40}$$

and the function $f : \mathcal{M} \to \mathfrak{g}$ is now,

$$\begin{aligned} f(\lambda_p(u)) &= \overline{\Psi(u(t))pI^{-1}p^{-1}\Psi(u(t))^{-1}L^s} \\ f(A(t)) &= \overline{A(t)I^{-1}A(t)^{-1}L^s} = \overline{\omega^s(t)}. \end{aligned} \tag{4.41}$$

Recall that in the earlier section where we derive the differential equation for the rotation matrix $A(t)$ in Eq (4.18). Now, we can substitute Eq (4.41) into this, so that Eq (4.18) is defined in terms of the matrix $A(t)$ which we can solve numerically,

$$\dot{A}(t) = \overline{\omega^s(t)}A(t) = f(A(t))A(t). \tag{4.42}$$

Hence, the differential equation $\dot{u}(t)$ in Eq (4.39) can be thought of as one which originated from the differential equation Eq (4.42) evolving on the Lie group $A \in \mathrm{SO}(3)$ [8].

In summary, when numerically solving the rigid body equation in free motion using the RKMK integrator, we want to solve $\dot{u}(t)$ in Eq (4.39) in $\mathfrak{so}(3)$ instead of solving the two differential equations given by Eq (4.20) and the Euler Equation in Eq (4.27) directly (as one would do if using a classical integrator like the Runge-Kutta method.) This is because the Euler Equations in Eq (4.27) can be transformed into its generic form given by Eq (4.42). By a change of variables in terms of the Lie group action in Eq (4.35), the differential equation $\dot{A}(t)$ evolving on $G = \mathrm{SO}(3)$ can be further transformed into one in terms of $u(t)$ evolving on the $\mathfrak{so}(3)$, given by Eq (4.39).

Once $u(t) \in \mathfrak{so}(3)$ is found by the RKMK integrator, we can then find the corresponding rotation matrix $A(t) \in \mathrm{SO}(3)$ by applying the Lie group action in Eq (4.35), which transform $u(t) \in \mathfrak{so}(3)$ to $A \in \mathrm{SO}(3)$, and hence preserving the orthogonality of the new rotation matrix $A(t)$ in each time step.

Assuming that at $t = 0$, the body and space coordinates coincide, then the angular momentum in the space coordinate L^s is a constant vector and is defined by the initial angular momentum $L^b(0)$ of the rigid body in body coordinates. Similarly, the position vector r^b is a constant vector with respect to the body coordinate, and is defined by the initial position $r^s(0)$ in the space coordinates. Once the rotation matrix $A(t_{n+1})$ for time step t_{n+1} is found, we can then find the corresponding values of angular momentum $L^b(t_{n+1})$, angular velocity $\omega^b(t_{n+1})$, and position $r^s(t_{n+1})$of the rigid body by using Eq (4.32), Eq (4.13), and Eq (4.15),

$$\begin{aligned} L^b(t_{n+1}) &= A(t_{n+1})^{-1}L^s, \\ \omega^b(t_{n+1}) &= I^{-1}L^b(t_{n+1}), \\ r^s(t_{n+1}) &= A(t_{n+1})r^b. \end{aligned}$$

4.5.7 Numerical Results

In this section we will implement the RKMK integrator to solve the free rotation of the rectangular block defined in Section 4.5.5. Also, for comparison, we solved for $\omega^b(t)$ and $r^s(t)$ using traditional integrators such as the Runge-Kutta method, the implicit Gauss method, and the build-in ODE45 in Matlab. With the exception of ODE45, all other integrators are of 4th order and are implemented with fixed stepsize $h = 1/40$.

The Runge-Kutta method (RK4), and one used by the RKMK integrator, is based on the following explicit Butcher tableau,

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array}$$

In the RKMK integrator, two canonical coordinate maps of the first kind $\Psi : \mathfrak{g} \to G$ have been implemented. These are the exponential map and the Cayley map, denoted RKMK4(exp) and RKMK4(cay). Note that for the exponential map, the Matlab function **expm** was used. In the differential equation $\dot{u}(t)$ in Eq (4.39), $d\Psi^{-1}_{u(t)}$ is given by the inverse of the differential of the exponential map in Theorem 1.3, where, for $u, v \in \mathfrak{so}(3)$,

$$\mathrm{dexp}_u^{-1}(v) = \sum_{k=0}^{\infty} \frac{B_k}{k!}\mathrm{ad}_u^k(v).$$

For the Cayley map, the following inverse was used,

$$\mathrm{dcay}_u^{-1}(v) = v - \frac{1}{2}\mathrm{ad}_u(v) - \frac{1}{4}uvu.$$

Both coordinate maps were used in the RKMK framework to solve Eq (4.39) for $u(t) \in \mathfrak{so}(3)$, then $A(t) \in \mathrm{SO}(3)$ via the Lie group action. After that, we advance the solutions $L^b(t)$, $\omega^b(t)$, and $r^s(t)$ as stated in the previous section.

It is interesting to note that the rigid body in free rotational motion has the Hamiltonian in Eq (4.28) which is a quadratic function. It is well known that symplectic methods such as the Gauss methods, and the special partitioned Runge-Kutta methods are able to preserve quadratic invariants. For comparisons, the following 2-stage, 4th order Gauss method below, discussed in

Chapter 2, has been implemented to solve the Euler Equations in Eq (4.27) for $\omega^b(t)$ and Eq (4.20) for rotation matrix $A(t)$.

$$\begin{array}{c|cc}
\frac{1}{2}-\frac{\sqrt{3}}{6} & \frac{1}{4} & \frac{1}{4}-\frac{\sqrt{3}}{6} \\
\frac{1}{2}+\frac{\sqrt{3}}{6} & \frac{1}{4}+\frac{\sqrt{3}}{6} & \frac{1}{4} \\
\hline
 & \frac{1}{2} & \frac{1}{2}
\end{array}$$

Because the method above is implicit and the problem is not stiff, a fixed-point iteration scheme is used. The convergence of the internal stages Y_1 and Y_2 defined by the Gauss tableau is achieved in two different ways. One way is the iteration scheme based on the Gauss-Seidel method (denoted Gauss(g-s)), which has a linear convergence. Since the rigid body problem is a non-stiff problem, this method should work with reasonable accuracy. The other iteration scheme is a Full-Newton update (denoted Gauss(f-n)). This is computational expensive since the Jacobian matrix is updated at every step, but this approach has a quadratic convergence. Again, because the integration period in the following experiments are relatively short, hence a full-newton update should not cause great problems.

Firstly, we checked the order of the RKMK4 integrators by calculating the error ratios for the angular velocity $\omega^b(t)$ and the position vector $r^s(t)$ when the stepsize h was halved. The results are displayed in the two log-log plots in Figure 4.13, and the numerical values of the error ratios for ω^b and r^s can be found in Table A.6 for RK4, Table A.7 for RKMK4 with the exponential map, and Table A.8 for RKMK4 with the Cayley map, all in Appendix A.

By checking the error ratios in the tables, it is true that all three integrators are of order 4. It is also interesting to note that the classical RK4 integrator produced the most accurate results with the larger stepsize h for the angular velocity $\omega^b(t)$ on the left diagram in Figure 4.13, but the same method produced the least accurate $r^s(t)$ solution on the right hand side plot in comparison to the other two RKMK integrators. The former might be due to the accumulating round-off error in the great number of evaluations that one step of an RKMK method must do in comparison the relatively few evaluations in a single step of an RK method. However, since RK method does not preserve the orthogonality of the rotation matrix $A(t)$, then the error from this method is greater in each step than the other two RKMK integrators.

Along with other considerations such as computational costs and accuracy, a good integrator should also be able to produce solutions which display well known characteristics from qualitative analysis of the differential system. For example, numerical solutions produced from the integration of the rotating rectangular block should demonstrate the stability of the rotation with respect to the three principal axes for 3 distinct moments of inertia, mentioned at the end of Section 4.5.3. By testing the RKMK integrator with the exponential map, we will see this by setting the rectangular block to rotate about its centre of mass in three slightly different ways, depending on the initial angular velocity $\omega^b(0)$ given.

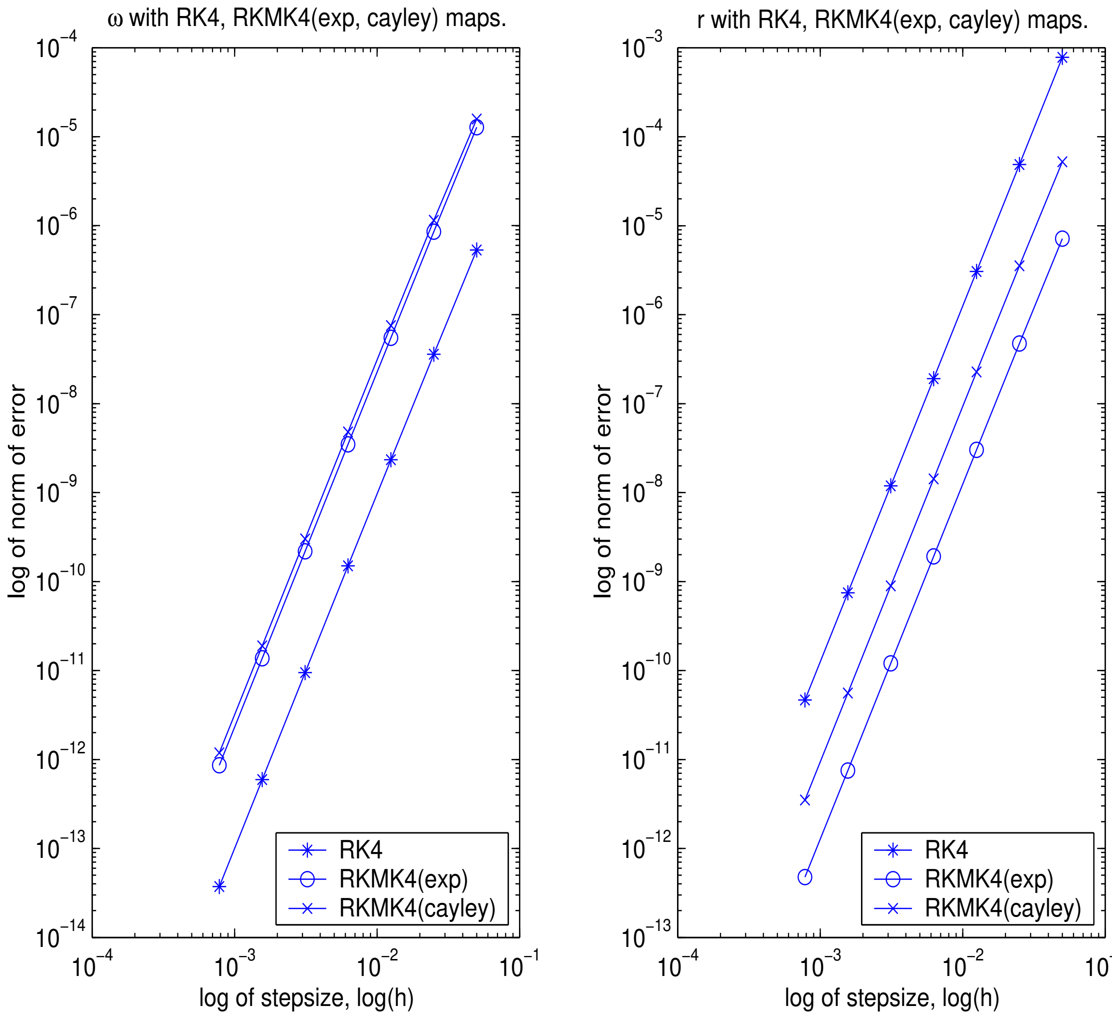


Figure 4.13: Log-log plot of global error against stepsize for the angular velocity $\omega^b(t)$ (left), and the position vector $r^s(t)$ (right), from solving the rigid body problem with the 3 methods RK4, RKMK4 with the exponential mapping, and RKMK4 with the Cayley mapping. $\omega^b(0) = [0.02, 5, 0.1]$, $t_0 = 0$, and $t_f = 1/10$.

Within Figure 4.14, Figure 4.15, and Figure 4.16, two diagrams are shown in each case. The diagrams on the left of the figures show the rotation of a corner of the rectangular block with respect to the whole rectangular block at the initial position $r^s(0)$. The diagram on the right shows 16 'snapshots' of the rectangular block rotating with its 3 body axes.

When the block is rotated precisely about any one of the principal axis, $\omega^b(t)$ is constant with respect to t. The rotating corner point from the RKMK method shows the circular orbit as time increases in Figure 4.14. This is the stable rotation of the rigid body, when set to rotate precisely about one of the principal axis.

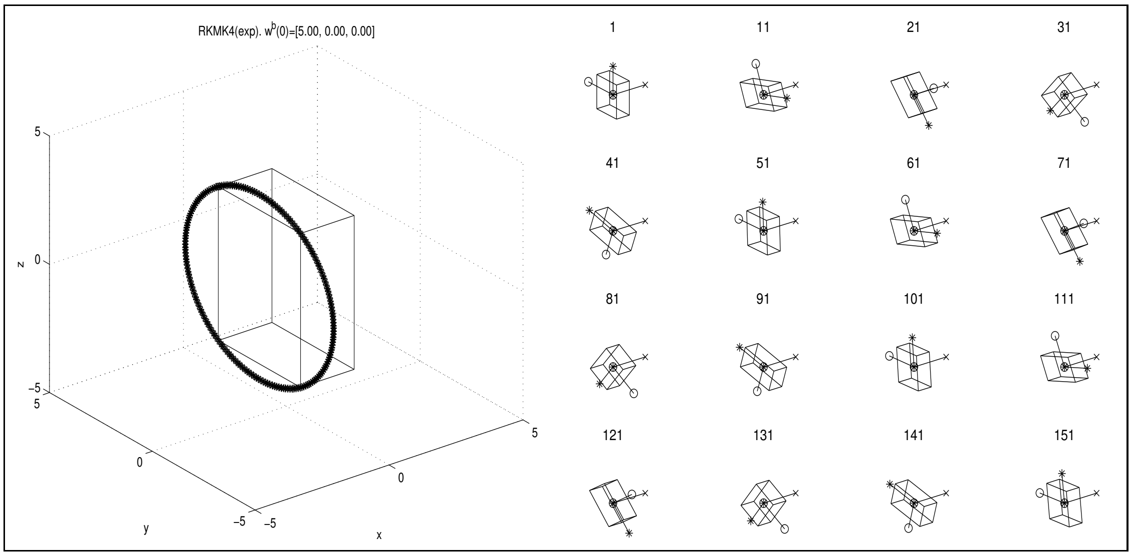


Figure 4.14: Plotting the rotation of one corner (left), and 3 dimensional rotation of the rectangular block, in the rigid body problem. It is rotating precisely about the x principal axis. $w^b(0) = [5, 0, 0]$, $t_0 = 0$, $t_f = 10$, and $h = 1/40$.

If the rectangular block is set to rotate 'roughly' about one of the principal axis corresponding to either the largest or the smallest moments of inertia, then the block appears to 'wobble' as it rotates about the chosen axis. (By 'roughly', we mean that the largest component of $\omega^b(0)$ is at the axis belonging to the largest or smallest moment of inertia.) This can be seen in the two diagrams in Figure 4.15, where the block is set to rotate about the principal x axis with the largest moment of inertia. In this case, from the left hand side diagram, the block is no longer rotating in a smooth circular orbit, but rotating only roughly about the original principal x axis, and the angular velocity $\omega^b(t)$ oscillates with time t. The largest component of $\omega^b(t)$ is the x component.

Lastly, to demonstrate the unstable motion of the rigid body, the block is set to rotate roughly about the principal y axis which corresponds to the second moment of inertia I_2, where $I_1 > I_2 > I_3$. Because the angular velocity ω^b_x and ω^b_z can grow exponentially with time t in the oscillation, then those two components can dominate the solution such that the block is no longer rotating about the original y axis. In fact, the rectangular block now appears to be 'tumbling' in all 3 directions, as shown by the two diagrams in Figure 4.16.

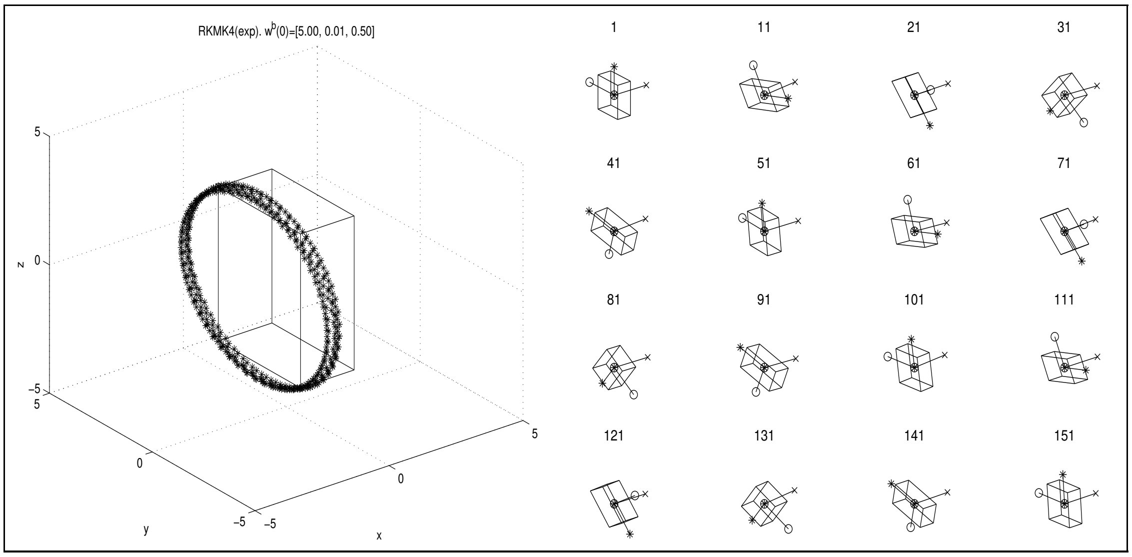


Figure 4.15: Plotting the rotation of one corner (left), and 3 dimensional rotation of the rectangular block, in the rigid body problem. It is rotating roughly about the x principal axis. $w^b(0) = [5, 0.01, 0.5]$, $t_0 = 0$, $t_f = 10$, and $h = 1/40$.

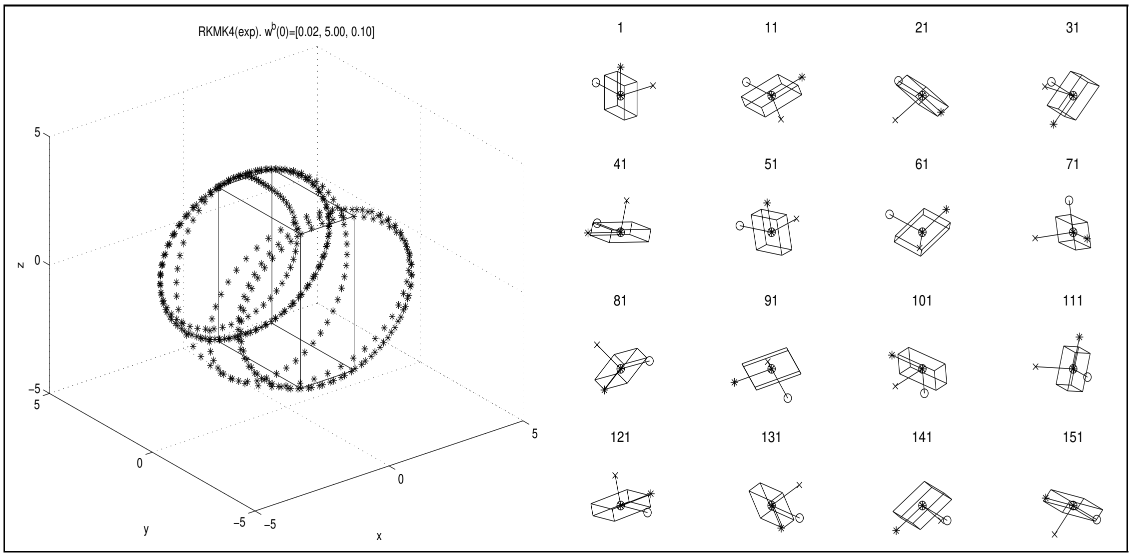


Figure 4.16: Plotting the rotation of one corner (left), and 3 dimensional rotation of the rectangular block, in the rigid body problem. It is initially rotating roughly about the y principal axis associated with I_2, causing unstable motion of the rigid body. $w^b(0) = [0.02, 5, 0.1]$, $t_0 = 0$, $t_f = 10$, and $h = 1/40$.

Next, we will see how the performance of the RKMK integrator (with the exponential map), compares with that of the RK method. We started with an initial angular velocity $\omega^b(0) = [0.02,\ 5,\ 0.1]$ and integrating over the interval $t \in [0, 40]$ with a constant stepsize of $h = 1/40$. From the two integrators, the results are used to plot four graphs in Figure 4.17.

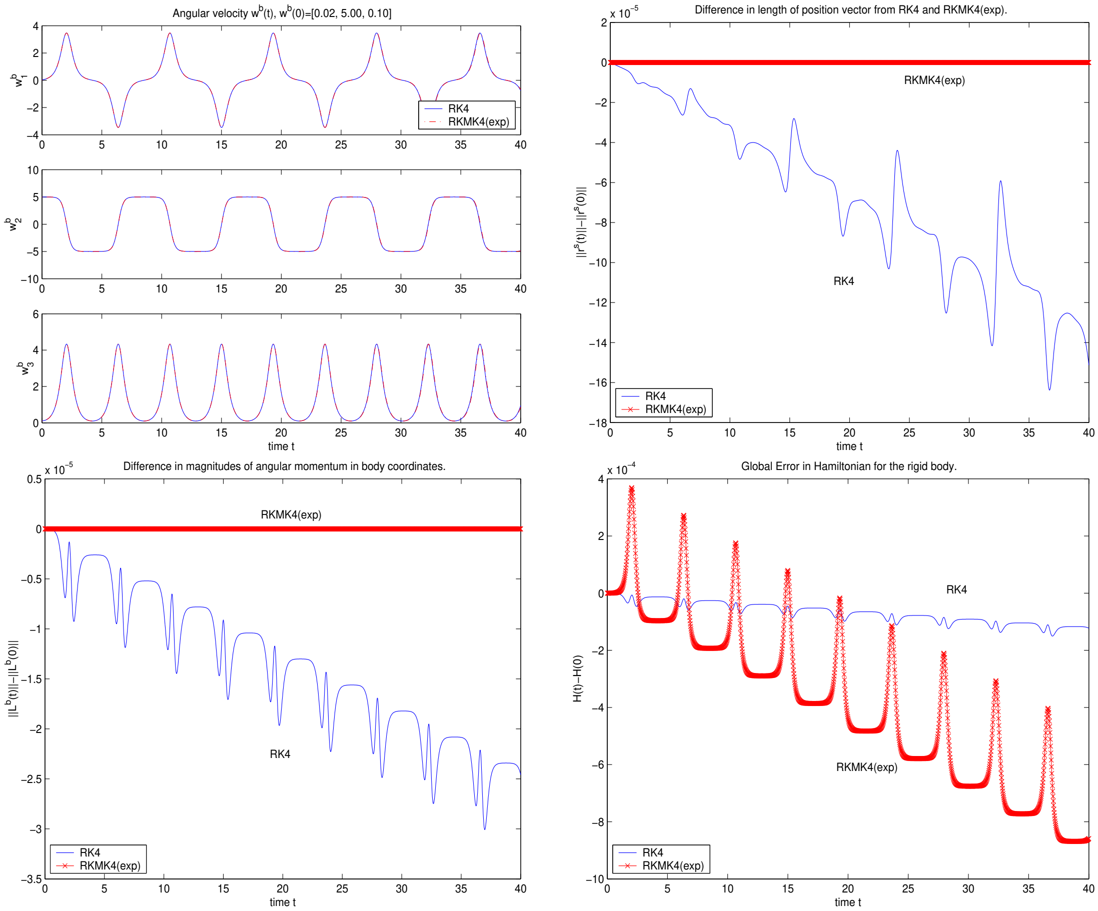


Figure 4.17: Results from RK4 and RKMK4(exp) when solving the rigid body problem. The 3 components are angular velocity $\omega^b(t)$ in body coordinates (top left), the norm of the position vector from 1 corner of the rotating rectangular block (top right), the error in the norm of the angular momentum in body coordinates $\|L^b(t)\|$ (bottom left), and the error in the Hamiltonian $\|H(t)\|$ (bottom right). $w^b(0) = [0.02,\ 5,\ 0.1]$, $t_0 = 0$, $t_f = 40$, and $h = 1/40$.

The graph on the top left hand corner of Figure 4.17 shows the 3 components of the angular velocity $\omega^b(t)$. From this plot, there are no visible difference between the solutions from RK4 and that of the RKMK4 with the exponential mapping. However, the graph on the top right hand corner, which displays that norm of the position vector $\|r^s(t)\|$ for one corner of the rectangular block, shows a decrease in magnitude from its original length (approximately 10^{-5}), for solutions from RK4 method, while solutions from RKMK4 preserves this norm. This is because when solving for the rotation matrix $A(t) \in \mathrm{SO}(3)$ from Eq (4.20) using RK4, this integrator is not designed to preserve this orthogonality, so in the resultant rotation of $r^s(t)$, the magnitude of the vector is not preserved.

A similar behaviour can be observed for the magnitude of the angular momentum $\|L^b(t)\|$, which is plotted in the bottom left hand corner of Figure 4.17. The rotation matrix $A(t)$ produced by RKMK4 when solving Eq (4.39), is orthogonal to machine precision. Any update with this rotation matrix to find $L^b(t)$ and $\omega^b(t)$ will preserve length and angle of rotation. Hence, from the bottom left hand graph, we can see that $\|L^b(t)\|$ from RKMK4 with the exponential mapping has the same norm throughout the integration period, while once again, solutions from RK4 decreases in magnitude. This is one advantage of using Lie group methods over the classical methods with fixed stepsize h.

However, we can not conclude that RKMK4 performs better than RK4 without looking at the preservation of the Hamiltonian. It turns out that in this rigid body framework, the RKMK4 integrator is not designed to preserve the Hamiltonian of the system given in Eq (4.28). In fact, from the bottom right hand graph in Figure 4.17, the performance of the RKMK4 integrator is a lot worse than that of the RK4 method. Since the results from RKMK4 method has a decreasing oscillation as t increases, and might be a direct consequence of the results in $\omega^b(t)$, the next item to investigate is the accuracy of the numerical solution in a longer integration period.

Figure 4.18 resulted in an integration period $t \in [0, 400]$, of which the numerical angular velocity $\omega^b(t)$ in the second half of the integration period is displayed. For this experiment, three integrators are used, RK4, RKMK4 with the exponential map, and ODE45 from Matlab. In this case, ODE45 acts as a control, since the tolerance for convergence is set to be 10^{-14}, which should produce very accurate results. The idea is to compare the performance of RK4 and RKMK4(exp) integrators with that of the ODE45.

From Figure 4.18, the dark lines composed of results from both RK4 and ODE45, while the faint dotted lines represent results from RKMK4 with exp mapping. It is apparent that solutions from RKMK4 are out-of-phase with respect to those from ODE45. On a closer inspection, we can take the difference of the components of $\omega^b(t)$ at each t values in the integration period between RK4 (or RKMK4) and ODE45. The differences are displayed in Figure 4.19, where each columns in the figure are plots of the differences between results from ODE45 and other integrators under consideration. The 3 integrators tested here include RK4, RKMK4 with the exponential map, and RKMK4 with Cayley map. By observing the scales for ω^b on the diagrams, one can deduce that the difference between RK4 and ODE45 is minimal for the 3 components of $\omega^b(t)$, in comparison to the difference shown by RKMK4(exp) and RKMK4(cay). Hence it seems that the lack of accuracy in solutions from the RKMK integrators caused the loss of energy in the Hamiltonian calculations. Even though RKMK integrators are able to preserve $\|L^b(t)\|$ and $\|r^s(t)\|$ in the integration, but its inability to preserve the energy and its high computational costs renders it non-ideal for long term integration with fixed order q and stepsize h.

The importance of the preservation of energy can be displayed if we integrate this rigid body problem using Gauss method as well. As mentioned previously, Gauss method is a symplectic method, and will preserve quadratic invariants such as the Hamiltonian. Results from the two Gauss methods (differs in the iteration scheme used for the convergence of the internal stage values), are plotted along with results from the other integrators, RK4, RKMK4(exp), and RKMK4(cay) in Figure 4.20 and Figure 4.21.

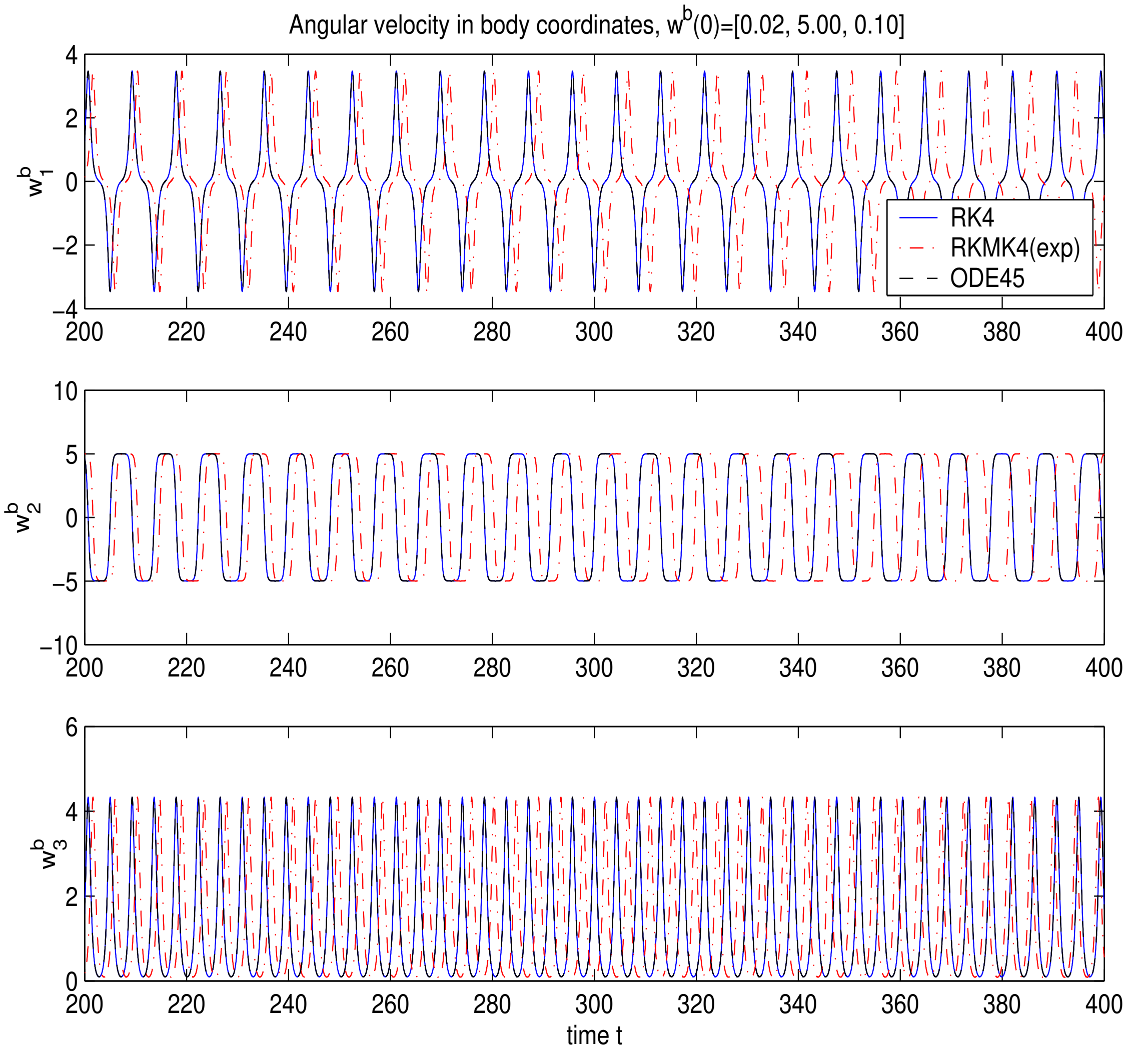


Figure 4.18: Results from RK4, RKMK4(exp), and ODE45 with a tolerance of 10^{-14}, when solving the rigid body problem. This is a plot of the 3 components of angular velocity $w^b(t)$, displaying results from the second half of the integration period from $t = [200, 400]$. $\omega^b(0) = [0.02,\ 5,\ 0.1]$, $t_0 = 0$, $t_f = 400$, and $h = 1/40$.

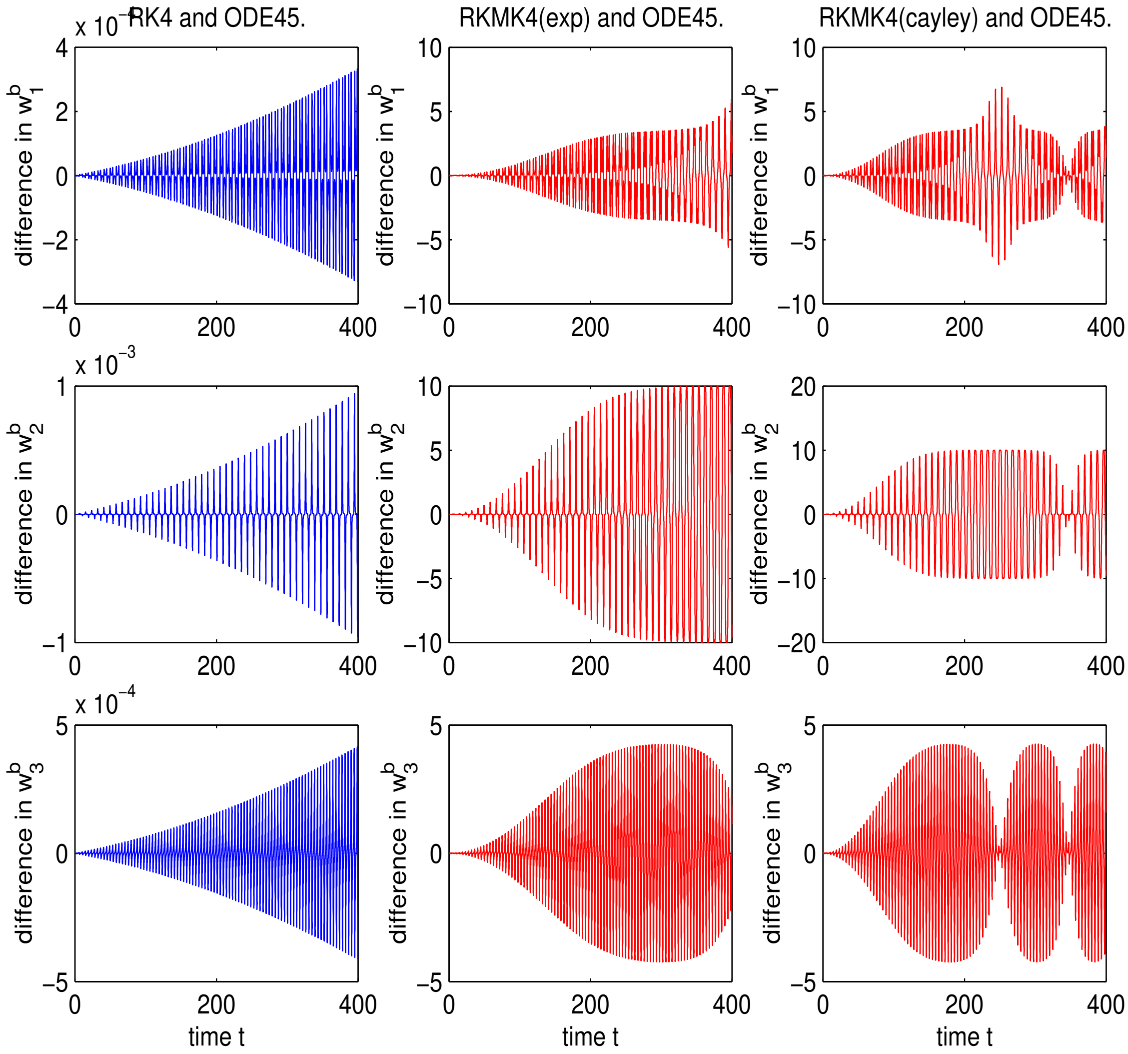


Figure 4.19: The difference between results from ODE45 and 3 other integrators, RK4, RKMK4(exp), and RKMK4(cay) for discretized values of t in the integration period, when solving the rigid body problem. $\omega^b(0) = [0.02, 5, 0.1]$, $t_0 = 0$, $t_f = 400$, and $h = 1/40$.

The left hand graph in Figure 4.20 shows the disadvantage of using RK4 method as it does not preserve the norm of the angular momentum $\|L^b(t)\|$. The right hand graph shows the difference between the 4 other integrators, which appear to give the same results if only the left hand graph is observed. From this right hand graph, the two Gauss methods performed better than the RKMK integrators given that the differences in solutions from Gauss methods are closer to zero in this comparison.

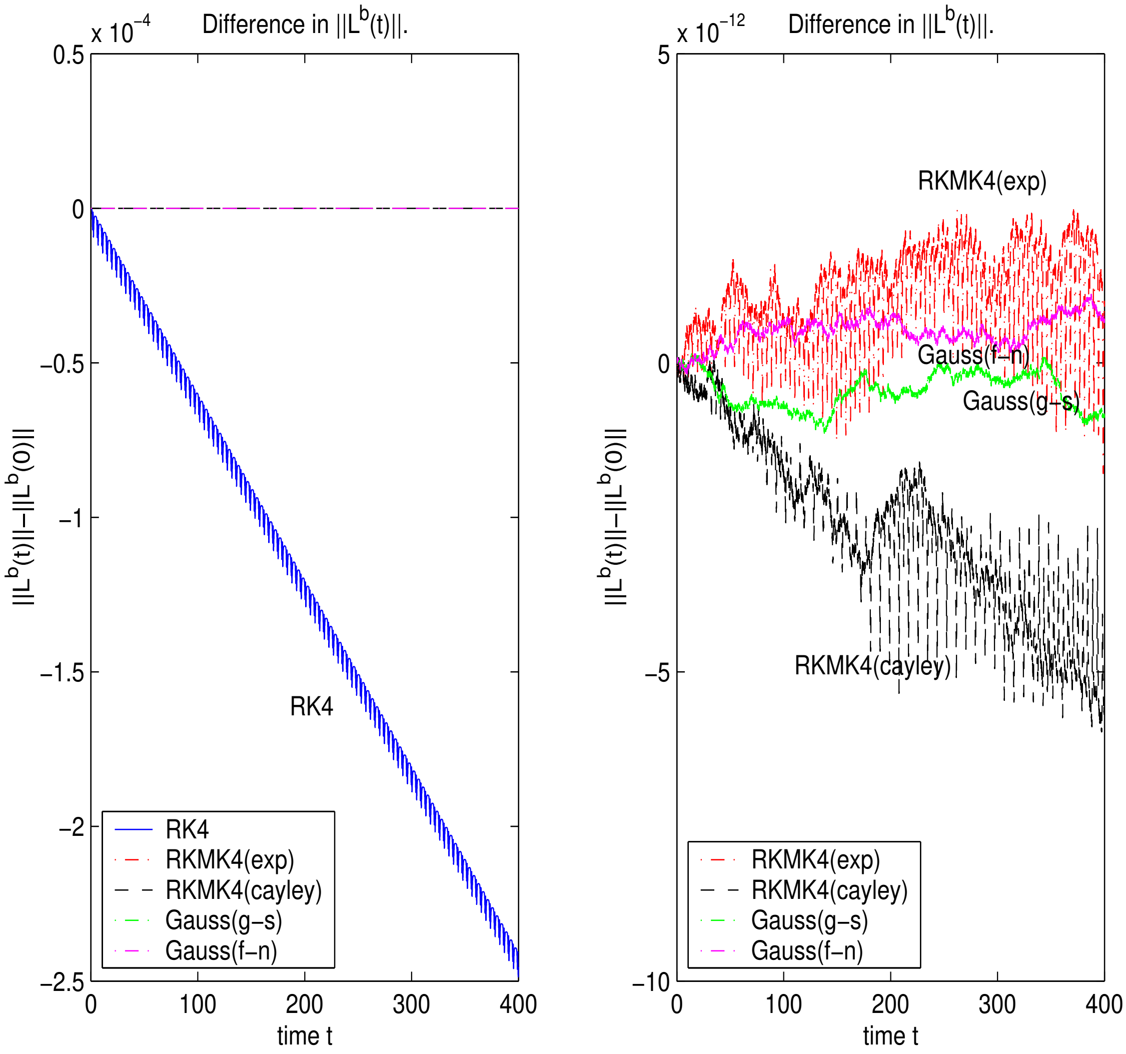


Figure 4.20: The error in the norm of the angular momentum in body coordinates $\|L^b(t)\|$ against time t, from RK4, RKMK4(exp), RKMK4(cay), GAUSS(using Gauss-Seidel iterations), and GAUSS(using Full-Newton iteration), when solving the rigid body problem. The graph on the left hand side shows the results from all the integrators listed. Note that on the left hand graph, results from all of the integrators except the RK4 method are very close together along 10^{-4}. The right hand graph is a scaled version of the left hand graph. $\omega^b(0) = [0.02,\, 5,\, 0.1]$, $t_0 = 0$, $t_f = 400$, and $h = 1/40$.

In Figure 4.21, the difference in the Hamiltonian of the numerical solutions are plotted. It is clear that the two Gauss integrators performed best in this experiment, while the RKMK method are no good when the preservation of energy is under consideration. It is also of interest to note that the severity of this loss of energy from the RKMK integrators are dependent on the coordinate maps used. From implementing the canonical coordinates of the first kind for this rigid body problem, it is clear that the Cayley mapping is worse than the exponential mapping. This observation has also been noted by Owren and Celledoni in [8].

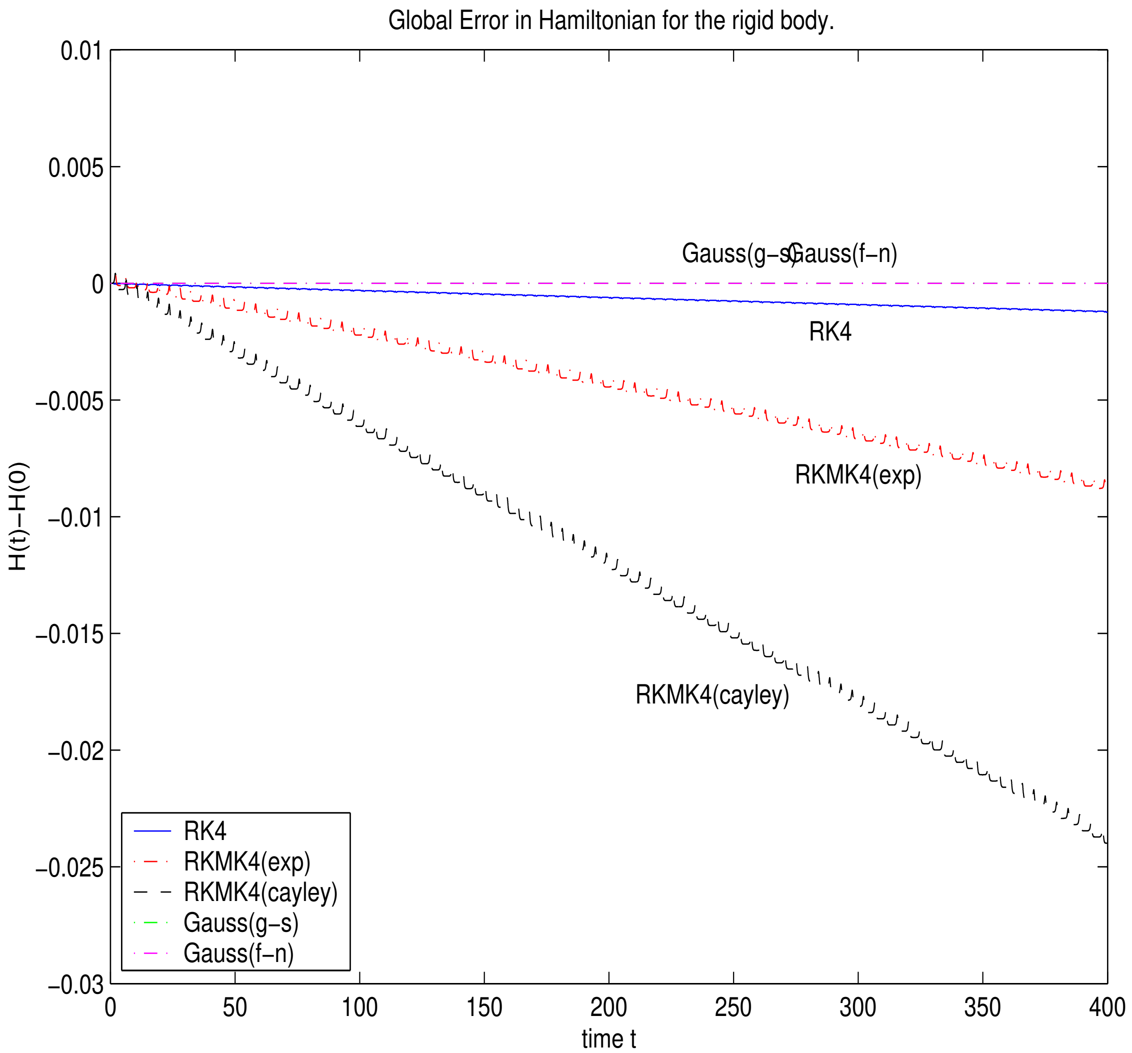


Figure 4.21: The error in the Hamiltonian $\|H(t)\|$ against time t, from RK4, RKMK4(exp), RKMK4(cay), GAUSS(using Gauss-Seidel iterations), and GAUSS(using Full-Newton iteration), when solving the rigid body problem. Note that the results from both Gauss(g-s) integrator and the Gauss(f-n) are very close to each other, so they appear as a single straight line across 0 on the graph. $\omega^b(0) = [0.02, 5, 0.1]$, $t_0 = 0$, $t_f = 400$, and $h = 1/40$.

CHAPTER 5

Conclusion

The first section of this chapter surveys the work in the main body of this thesis. The second section contains other Lie group methods which have not been mentioned in the discussion. Finally, we conclude the thesis with a survey of possible future work in the study of Lie group methods in geometric integration.

5.1 Lie group methods

Chapter 1 introduced the basic language of manifolds, Lie groups and Lie algebras. It also covered other important mathematical concepts such as Lie derivatives, pull-backs, Lie group actions, and the differential of the exponential map. The generic presentation of differential equations was discussed, and this key idea was used in the derivation of results concerning the Runge-Kutta-Munthe-Kaas method in Chapter 3.

Chapter 2 presented a discussion on the basic development of the Magnus series expansion on the linear matrix differential equation. It traced the steps toward deriving the expansion of $\Omega(t)$ using Picard iterations, and showed the need to introduce the rooted tree theory, which allows for the computation of terms of high order with relatively little effort. Also, the idea that time-symmetry can be used to reduce the number of terms in the Magnus series expansion of up to a particular order was mentioned. In particular, if the truncated series is expanded up to an odd power of terms, then there is a gain of an extra order in the overall expansion of $\Omega(t)$. Finally, the idea of multivariate quadrature was discussed. This section in Chapter 2 provided a brief summary of what is involved when applying the Gauss-Legendre quadrature to numerically integrate each term in the Magnus series expansion.

The Runge-Kutta-Munthe-Kaas (RKMK) method was introduced in Chapter 3, and is one of the main classes of Lie group methods considered in this thesis. It is an important class of methods, because it allows the use the classical Runge-Kutta methods, while preserving the group structure of the manifold associated with a particular dynamical system.

In the first part of Chapter 3, the concepts of an equivariant map and relatedness between vector fields were presented. Such ideas allows us to consider an equivalent differential equation which evolves on the Lie algebra $\mathfrak{g}$, instead of the original differential equation evolving on the manifold $\mathcal{M}$. The advantage of this approach is that a classical Runge-Kutta method can be applied to solve the equivalent differential equation on $\mathfrak{g}$, as it is a linear vector space. After the approximation is made in $\mathfrak{g}$, the corresponding element on the manifold $\mathcal{M}$ can be found using the Lie group action, and thereby preserving the group structure in the numerical solution. The RKMK Algorithm was presented in Section 3.3, and this allows for the construction of the RKMK integrator in any scientific computer language.

The second part of Chapter 3 dealt with the derivations of the order conditions associated with the RKMK method. The Lie-Butcher series allows for the expansion of the analytical solution, as well as the numerical solution on the manifold. Their coefficients can then be compared, in a similar way as for the classical methods, and the order conditions for an arbitrary order of a RKMK method can be derived. It turned out that the order conditions for the RKMK method are the same as those from the classical method, and hence the RKMK method has the same order q as the classical RK method on which it is based.

Chapter 4 contained five numerical experiments integrated by the two main classes of Lie group methods, the Magnus series expansion and the RKMK method, introduced in the previous chapters. Observations were made regarding their performance in comparison to the classical integrators such as the Runge-Kutta method, and the Gauss method, all of the same order q and constant stepsize h.

The first two experiments involved the integration of the harmonic oscillator and of the three dimensional Lorenz equations. The former was solved using the Magnus series expansion, and the latter was designed to demonstrate how we can reduce the RKMK structure to the familiar form of the classical Runge-Kutta method. With the harmonic oscillator, the Magnus series expansion worked favourably when compared with both the Runge-Kutta and the Gauss methods. Meanwhile, due to the sensitive dependence on initial conditions, integrating the Lorenz equations showed clearly that the classical Runge-Kutta method of fixed order and stepsize, was not the ideal integrator to use when solving this particular dynamical system.

The third experiment in Chapter 4 was designed to show the advantage of using the RKMK method over the other classical integrators. The solution $y(t)$ is an element of the Special Orthogonal matrix Lie group SO(3), and f is a map from SO(3) to its corresponding Lie algebra $\mathfrak{so}(3)$. As expected, the RKMK method successfully preserved the orthogonality and unit determinant of the numerical solution $y(t)$ over the integration period, but numerical solutions from the classical RK method failed to meet both requirements. Also, in this experiment, the issues concerning the accuracy of the matrix exponential calculation, and the algorithm on which the Matlab function file **expm** was based, were studied. This lead to the application of the compensated error addition technique, which can be used to approximate the rounding error, and briefly overcome its accumulation when the limit of machine precision has been reached in the calculation. It was discovered that the compensated error technique can not only be applied to the classical scheme, but also successfully to the RKMK method as well.

The fourth experiment in Chapter 4 was a differential equation which evolves on the sphere, and was solved by both the Magnus series expansion and by the RKMK method. This particular example again demonstrated that the Lie group methods produce numerical solutions which adhere to the manifold, while those from the classical Runge-Kutta method eventually drifted away.

The final experiment in Chapter 4 was the integration of the rigid body equations in free motion. Given a particular initial angular velocity $\omega(t_0)$ and initial position $y(t_0)$ of the object under free rotation, numerical integration can be done to find its angular velocity $\omega(t)$ and its position $y(t)$ at a given time $t > t_0$. In this case, the Lie group G is SO(3), its corresponding Lie algebra is $\mathfrak{so}(3)$, and the manifold $\mathcal{M}$ can be defined as the dual space $\mathfrak{so}(3)^*$, where the angular momentum of the rigid body, $L^b(t)$ is a curve in this dual space. Then, the RKMK integrator with fixed order and fixed step, successfully preserved the norm of the angular momentum $\|L^b(t)\|$, as well as the norm of the position vector $\|y^s(t)\|$. On the other hand, the same RKMK integrator was not able to preserve the numerical Hamiltonian, and this is believed to be a direct consequence of the inaccuracy caused by the eventual drifting of the angular velocity $\omega(t)$ from the true solution. As yet, the reason for this effect is unknown, and it remains an open problem to be solved when applying the RKMK method to integrate the rigid body problem in free motion.

5.2 Future Work

This thesis, in many respect, is only a brief summary of some of the basic ideas which have gone into the theoretical and computational aspects of the Lie group methods. This section deals with some of the possible future work which I would like to explore further. Most of the information below are from various references provided in the bibliography, and in particular, from the extremely useful paper in [19], which provides a comprehensive overview on the subject of Lie group methods.

One major development of the Lie group theory which has not been included in this thesis, is the pioneering work of Crouch and Grossman. The Crouch-Grossman method approximates the steps in the numerical integration in the Lie group G, which is in general more complicated than the RKMK scheme of approximating each step in the linear Lie algebra $\mathfrak{g}$. It has also been proved that the rooted tree theory used in the classical order analysis is possible for the Crouch-Grossman method, and that there is no fourth order method of this kind with four stages, as is possible with the RKMK methods.

In Chapter 2, the Magnus series expansion was introduced to solve the linear differential equation in Eq (2.1), where the matrix A is dependent on time t only. However, when $A = A(t, y)$, then the problem becomes nonlinear and collocation methods must be used to solve such differential equations. This approach has been extensively developed, and it is something that I hope I may be able to study in more detail, in the future.

With regards to the RKMK class of integrators, an aspect which has not been explored in depth here, is the use of other canonical coordinates. It was mentioned in Chapter 3 of this thesis, that the exponential map and the Cayley map are called canonical coordinates of the first kind. The canonical coordinates of the second kind are more complicated, and involve the replacement of one single exponential or Cayley mapping, with a product of exponential or Cayley mappings, based on the linear combination of the basis in the Lie algebra $\mathfrak{g}$. The advantage of this method is the surprising reduction in computational cost, since the matrix exponential of each element in the basis can be relatively cheap to compute. However, due to the complexity of this specialist topic, it was deemed best to omit this discussion at present, and defer this to possible future study.

Quantification of the advantage Lie group methods have over the traditional methods is also another very interesting, but difficult subject to cover, and in practise it is hard to implement fairly and consistently. There are many attempts at quantifying the performance of numerical methods. One such attempt is the comparison of computational cost involved when using a particular integrator to solve differential equations over a certain period of time. In the numerical experiments performed in Chapter 4, no computational cost calculations have been done, mainly because a small section in the thesis cannot do justice to such a complicated issue. At present, it remains unclear what approach is most suited for measuring computational cost for Lie group methods in a fair and consistent manner, and the discussion below provides preliminary ideas about what measurements might or might not be suited for the comparisons of such integrators.

Traditionally, people use CPU time, which depends on the number of operations to be done, as a measure of cost for a particular integrator. However, this type of measurement is not the most reliable comparison, since the performance of the computer one uses to perform the integration can vary significantly. Another measure of computational cost is the counting of the number of function evaluations required in an integration. In this case, especially for the RKMK method, it is important to distinguish which is the 'function' under consideration, and what are the 'overheads' involved. Suppose that the differential equation has the following simple form,

$$\dot{y}(t) = F(y) = f(y)y, \qquad \text{where } y \in G, \quad f : G \to \mathfrak{g}.$$

Then, if the map $f : G \to \mathfrak{g}$ is the function to be counted for the cost calculations, at first glance it would seem that for fixed order and constant stepsize, both the RKMK method, and the classical RK method on which the Lie group method is based, have the same number of function evaluations for any given step in the integration. But this observation is misleading, because when a simulation is done with both methods on the same computer, it becomes very obvious that the CPU time taken for the RKMK integrator is significantly longer than for the RK integrator. This is due to the relatively large increase in the overhead cost of the RKMK scheme in comparison to the RK scheme. Overhead cost includes the number of operations taken to calculate the matrix exponential **expm** in Matlab, for example, or the extra operations required for evaluating the expression dexp^{-1} which contains a sum of the adjoint operators ad^k of kth degree. Such details are hidden inside each internal stage calculations, in one step of the RKMK method. Furthermore, such extra calculations are not required for the classical RK method, which makes the overhead cost for the RKMK method so much higher than their classical counterpart. Another basic method of cost calculation, is the counting of the number of multiplications required by each integrator. But this type of measurement faces the same difficulties as counting 'overhead' costs, simply because the RKMK method will require an overwhelming large number of multiplications in comparison to the RK method. Such comparisons show the computational inefficiency of the RKMK method, with no weighing to the possible gain in the qualitative behaviour of solutions. Hence, the fairness of such comparison remains uncertain.

This leads to the overall difficulties in making comparisons on the performances of integrators, and especially in evaluating the effectiveness of Lie group methods over a wide range of problems. It has always been difficult to make realistic comparisons between various numerical methods, because each integrator has its own individual strengths and weaknesses. With the classical approach, integrators are constructed to solve either stiff or non-stiff differential equations, and there exist a series of 'test' problems, such as DETEST, which ranges over different levels of difficulties. One can then evaluate the performance of the integrators on some common 'level', and comment on the advantages and efficiency of different methods. However, such an attempt at making comparisons for Lie group methods, or other geometric integrators, is more difficult. This is because integrators in this class are all designed to solve particular types of problems, not the wide variety of differential equations which most classical methods can solve. In some cases like the RKMK class of integrators, the main assumption is that you can write the original differential equation in terms of infinitesimal generators with respect to group actions. This assumption implies that not every differential equation can be integrated by the RKMK scheme, and care must be taken to identify all of the components like the manifold $\mathcal{M}$, the Lie group G and Lie algebra $\mathfrak{g}$ in a particular system, before any RKMK integrator can be constructed. For some differential equations, this task might not be possible. The heart of the problem lies in finding a way to classify types of differential equations which can be solved by Lie group methods, or more generally, by other types of geometric integrators. One believes that when such classification is clearly defined, then the construction of 'tests' for a fair and consistent comparison of performance of Lie group methods against the traditional methods will be possible. At present, however, such a classification of differential equations remains unclear, and this is another possible future topic which would be very interesting to explore further.

With regard to the high computational costs of Lie group methods, such as the RKMK scheme, work is ongoing to reduce the number of commutators in the integration process when using such methods. In particular, the idea of using Free Lie algebras is an attempt to make such integrators more efficient and competitive in long-term integrations. This, among many others, is the technical detail in the construction of Lie group methods which is currently outside the scope of this thesis.

Another area which would be interesting to explore further is the application of Lie group methods to systems of differential equations, other than the examples included in Chapter 4. In particular, it would be interesting to construct Lie group methods solely for the retention of the symplectic form, or the Lie-Poisson structure, or the conservation of energy and so on, when they are applied to integrate the Hamiltonian problems over a long period of time. From the preliminary work on the rigid body problem, it was observed that the RKMK method can preserve the angular momentum of the system over the entire integration period. But what remains unclear is how to structure the differential equation, or the Lie group method, such that the energy of the system can also be preserved as well.

Finally, there exist many geometric integrators other than Lie group methods. It is my hope that some of my future work will allow me to study the theory and applications of a wider range of geometric integrators beyond the Lie group methods.

APPENDIX A

Tables of Results

This appendix contains tables of results from the numerical experiments in Chapter 4.

Table A.1: Global errors from halving of stepsize h when solving Eq (4.2), with $t_0 = 0$ and $t_f = 0.1$. The top table shows the error between each subsequent halving of h, while the bottom table contains the ratios of the global errors. All integrators used are of order 4.

h	RK $\|y_i - y_{i-1}\|$	Gauss $\|y_i - y_{i-1}\|$	Magnus $\|y_i - y_{i-1}\|$
1/10	0.000000000000000	0.000000000000000	0.000000000000000
1/20	0.000000176274909	0.000000028889231	0.000000001169750
1/40	0.000000011022878	0.000000001806441	0.000000000073055
1/80	0.000000000689099	0.000000000112916	0.000000000004565
1/160	0.000000000043074	0.000000000007057	0.000000000000286
1/320	0.000000000002693	0.000000000000442	0.000000000000017
1/640	0.000000000000168	0.000000000000027	0.000000000000003
1/1280	0.000000000000011	0.000000000000001	0.000000000000012
1/2560	0.000000000000002	0.000000000000001	0.000000000000021
1/5120	0.000000000000006	0.000000000000006	0.000000000000036
1/10240	0.000000000000011	0.000000000000011	0.000000000000090
1/20480	0.000000000000016	0.000000000000016	0.000000000000429
h	ratio	ratio	ratio
1/10	0.000000000000000	0.000000000000000	0.000000000000000
1/20	0.000000000000000	0.000000000000000	0.000000000000000
1/40	0.062532315171961	0.062529901897753	0.062453127024711
1/80	0.062515340132863	0.062507479822763	0.062482915358609
1/160	0.062508161779896	0.062499800503731	0.062717922810202
1/320	0.062510728212941	0.062616823060329	0.060756334192802
1/640	0.062222569112256	0.061517854134643	0.153322675223949
1/1280	0.063425053427839	0.051658722955070	4.663364351094303
1/2560	0.168467151379036	1.012422836565829	1.678683698851742
1/5120	3.283993816165826	4.134919706065616	1.726039918688145
1/10240	1.829481970285734	1.827043635242043	2.500329846767442
1/20480	1.508667905546973	1.510681342857626	4.757228046102076

Tables A.2, A.3, A.4, and A.5 contain results from solving Eq (4.5) in Section 4.3 from $t_0 = 0$ to $t_f = 10$, using integrators discussed in Section 4.3.3 and Section 4.3.4.

Table A.2: Results from original RK4 method.

h	$\det(y)$	$\|y^T y - \mathrm{Id}\|$
1/10	1.000000493241461674	0.000000519142239366
1/20	1.000000030606113643	0.000000032181705337
1/40	1.000000001905749336	0.000000002002863777
1/80	1.000000000118879795	0.000000000124911996
1/160	1.000000000007419843	0.000000000007800538
1/320	1.000000000000458744	0.000000000000489395
1/640	1.000000000000026423	0.000000000000040311
1/1280	0.999999999999985789	0.000000000000027768
1/2560	0.999999999999997780	0.000000000000014549
1/5120	1.000000000000004219	0.000000000000016227
1/10240	0.999999999999986677	0.000000000000037345
h	$\|y_i - y_{i-1}\|$	ratio
1/10	0.000000000000000000	0.000000000000000000
1/20	0.000000255176316519	0.000000000000000000
1/40	0.000000015779503260	0.061837648081977004
1/80	0.000000000981567290	0.062205208494120000
1/160	0.000000000061211399	0.062360879293989306
1/320	0.000000000003821891	0.062437574100442046
1/640	0.000000000000235924	0.061729538630575169
1/1280	0.000000000000020216	0.085688756084848106
1/2560	0.000000000000010779	0.533173489890559571
1/5120	0.000000000000019134	1.775194146172834664
1/10240	0.000000000000081722	4.270974508377147316

Table A.3: Results from RK4 method with compensated error addition.

h	$\det(y)$	$\|y^T y - \mathrm{Id}\|$
1/10	1.000000493241462118	0.000000519142239780
1/20	1.000000030606112755	0.000000032181704973
1/40	1.000000001905749780	0.000000002002864536
1/80	1.000000000118879573	0.000000000124912361
1/160	1.000000000007420287	0.000000000007801761
1/320	1.000000000000460965	0.000000000000490715
1/640	1.000000000000025979	0.000000000000034155
1/1280	0.999999999999998779	0.000000000000015879
1/2560	0.999999999999997113	0.000000000000017310
1/5120	0.999999999999996891	0.000000000000017370
1/10240	0.999999999999996891	0.000000000000017371
h	$\|y_i - y_{i-1}\|$	ratio
1/10	0.000000000000000000	0.000000000000000000
1/20	0.000000255176316766	0.000000000000000000
1/40	0.000000015779502709	0.061837645861787700
1/80	0.000000000981567551	0.062205227176095952
1/160	0.000000000061211003	0.062360459169132007
1/320	0.000000000003821474	0.062431165254867846
1/640	0.000000000000238738	0.062472626021634649
1/1280	0.000000000000014932	0.062546755154384862
1/2560	0.000000000000000879	0.058870075446163614
1/5120	0.000000000000000144	0.164255039578272949
1/10240	0.000000000000000002	0.012014106471437391

Table A.4: Results from original RKMK4 method.

h	$\det(y)$	$\Vert y^T y - \mathrm{Id}\Vert$
1/10	0.999999999999998113	0.000000000000017189
1/20	0.999999999999995892	0.000000000000015006
1/40	0.999999999999999778	0.000000000000010548
1/80	1.000000000000009770	0.000000000000022358
1/160	1.000000000000052403	0.000000000000083067
1/320	1.000000000000103473	0.000000000000163842
1/640	1.000000000000146105	0.000000000000222856
1/1280	1.000000000000329958	0.000000000000454542
1/2560	1.000000000000465628	0.000000000000611264
1/5120	1.000000000000206057	0.000000000000422054
1/10240	1.000000000000029754	0.000000000000134665
h	$\Vert y_i - y_{i-1}\Vert$	ratio
1/10	0.000000000000000000	0.000000000000000000
1/20	0.000000319885772459	0.000000000000000000
1/40	0.000000019888701971	0.062174387495568771
1/80	0.000000001239656387	0.062329677846438843
1/160	0.000000000077400803	0.062437303887825715
1/320	0.000000000004869064	0.062907154519425079
1/640	0.000000000000378646	0.077765731706521124
1/1280	0.000000000000366709	0.968474905634153060
1/2560	0.000000000000159826	0.435836914630514383
1/5120	0.000000000001281570	8.018557751657997557
1/10240	0.000000000001108748	0.865147958214525947

Table A.5: Results from RKMK4 method with compensated error addition.

h	$\det(y)$	$\Vert y^T y - \mathrm{Id}\Vert$
1/10	0.999999999999997002	0.000000000000017265
1/20	0.999999999999996891	0.000000000000017376
1/40	0.999999999999997002	0.000000000000017358
1/80	0.999999999999996891	0.000000000000017357
1/160	0.999999999999996891	0.000000000000017351
1/320	0.999999999999996891	0.000000000000017354
1/640	0.999999999999996780	0.000000000000017260
1/1280	0.999999999999996891	0.000000000000017364
1/2560	0.999999999999996891	0.000000000000017363
1/5120	0.999999999999997113	0.000000000000017343
1/10240	0.999999999999996891	0.000000000000017372
h	$\Vert y_i - y_{i-1}\Vert$	ratio
1/10	0.000000000000000000	0.000000000000000000
1/20	0.000000319885770523	0.000000000000000000
1/40	0.000000019888701512	0.062174386437444745
1/80	0.000000001239658894	0.062329805333896551
1/160	0.000000000077370751	0.062412935945371228
1/320	0.000000000004832275	0.062456095260376134
1/640	0.000000000000301933	0.062482645030955394
1/1280	0.000000000000018946	0.062749783336425877
1/2560	0.000000000000001057	0.055772929467278826
1/5120	0.000000000000000113	0.106683469276554976
1/10240	0.000000000000000167	1.480630637355962786

Tables A.6, A.7, and A.8 contain results from integrating the rigid body problem in Eq (4.26) and Eq (4.33) from Section 4.5.7. Solutions include the angular velocity $\omega^b(t)$ and the position vector $y^s(t)$ of the body, from $t_0 = 0$ to $t_f = 1/10$.

Table A.6: Results from RK4 method for the rectangular block in unstable free rotation, with $\omega(0) = [0.02, 5, 0.1]$.

h	$\|\omega_i - \omega_{i-1}\|$	ω ratio	$\|y_i - y_{i-1}\|$	y ratio
1/10	0.0000000000000	0.0000000000000	0.0000000000000	0.0000000000000
1/20	0.0000005341981	0.0000000000000	0.0007805607512	0.0000000000000
1/40	0.0000000359496	0.0672964169469	0.0000488239758	0.0625498729562
1/80	0.0000000023432	0.0651813310130	0.0000030521088	0.0625125008888
1/160	0.0000000001497	0.0638907264424	0.0000001907663	0.0625031128651
1/320	0.0000000000094	0.0632058726090	0.0000000119230	0.0625007731352
1/640	0.0000000000005	0.0628505000746	0.0000000007451	0.0625002316659
1/1280	0.0000000000000	0.0627883150187	0.0000000000465	0.0624998446941

Table A.7: Results from RKMK4 method with the exponential map for the rectangular block in unstable free rotation, $\omega(0) = [0.02, 5, 0.1]$.

h	$\|\omega_i - \omega_{i-1}\|$	ω ratio	$\|y_i - y_{i-1}\|$	y ratio
1/10	0.0000000000000	0.0000000000000	0.0000000000000	0.0000000000000
1/20	0.0000127871008	0.0000000000000	0.0000071497367	0.0000000000000
1/40	0.0000008545540	0.0668293846101	0.0000004733628	0.0662070334383
1/80	0.0000000549628	0.0643175469576	0.0000000302940	0.0639976072662
1/160	0.0000000034805	0.0633262954286	0.0000000019134	0.0631631136667
1/320	0.0000000002189	0.0628929827090	0.0000000001201	0.0628108906562
1/640	0.0000000000137	0.0626910039067	0.0000000000075	0.0626327761082
1/1280	0.0000000000008	0.0626173830510	0.0000000000004	0.0632666531886

Table A.8: Results from RKMK4 method with the Cayley map, for the rectangular block in unstable free rotation, $\omega(0) = [0.02, 5, 0.1]$.

h	$\|\omega_i - \omega_{i-1}\|$	ω ratio	$\|y_i - y_{i-1}\|$	y ratio
1/10	0.0000000000000	0.0000000000000	0.0000000000000	0.0000000000000
1/20	0.0000157731754	0.0000000000000	0.0000524015056	0.0000000000000
1/40	0.0000011461525	0.0726646659457	0.0000035486887	0.0677211226719
1/80	0.0000000752672	0.0656695299356	0.0000002268355	0.0639209516957
1/160	0.0000000047929	0.0636788676793	0.0000000142728	0.0629216444306
1/320	0.0000000003019	0.0629931455024	0.0000000008940	0.0626393428986
1/640	0.0000000000189	0.0627224960397	0.0000000000559	0.0625511619199
1/1280	0.0000000000011	0.0626267528711	0.0000000000034	0.0625366848153

APPENDIX B

Matlab Code

This appendix contains some of the Matlab codes written for the numerical experiments in the different sections of Chapter 4.

B.1 Harmonic Oscillators

The following Matlab code is the one step of an integrator based on the 4th order Magnus series expansion in Chapter 2, used to solve the differential equation in Eq (4.3).

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [tout, Yout] = magnus4(tin, Yin, h)
%magnus4.m
%function file for one step of a magnus series expansion to 4th order,
%on function file dY.m, to find Yout=[y; v].
%Amatrix.m is a function file containing A(t) matrix in dY.m.
%tin = input independent variable t at beginning of step.
%Yin = Y column vector at beginning of step. Y=[y; v]
%h = stepsize
%tout = output independent t value at the end of the step.
%Yout = Y column vector at the end of the step.

%Based on roots of the Legendre polynomial.
A1 = Amatrix(tin + (1/2-sqrt(3)/6)*h);
A2 = Amatrix(tin + (1/2+sqrt(3)/6)*h);
omega = 1/2*h.*(A1 + A2) + sqrt(3)/12*h^2.*(A2*A1-A1*A2);
Yout = expm(omega)*Yin;             %output Yout column vector and tout value.
tout = tin + h;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

To implement the function file `magnus4.m`, we implement it in another script file, similarly structured to the one displayed below.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all; close all; format long; clc;
t0 = 0;                                        %initial values.
Y0 = [-1;2];                                   %initial Y0=[y0;v0].
tf = 40;                                       %final value.
h = 1/20;                                      %fixed stepsize.
tspan = [t0:h:tf]';                            %vector of independent t values.

tout3 = tspan;                                 %Starting Magnus method.
Yout3 = Y0';
numstep=(tf-t0)/h;
t = tspan(1);                                  %initialize variables.
Y = Y0;
for i = 1:numstep
    [toutput, Youtput] = magnus4(t, Y, h);     %solve dY.m
    Yout3(i+1,:) = Youtput';                   %saving results.
    t = tspan(i+1);                            %update results.
    Y = Youtput;
end
clear numstep t Y i toutput Youtput;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

B.2 Lorenz Equations

In this experiment, we showed how to reduce the RKMK Algorithm into the classical RK method of order 4, and then used it to solve the Lorenz Equations in Section 4.2.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [tout, yout] = exprkmk4_exp(tin, yin, sigma, rho, beta, h)
%exprkmk4_exp.m
%function file for one step of an explicit rkmk4 method.
%lorenz.m is the function file containing the Lorenz Equations.
%Take M=G=g=R^3 in the classical case, so RKMK4 is reduced to RK4.
%action = addition, algebra is commutative, so that k_i=K_i in algorithm.
%tin = input independent variable.
%yin = column vector in R^3 at beginning of step.
%sigma, rho, beta are coefficients in the Lorenz Eqns.
%h = stepsize (fixed).
%tout = x value at the end of step.
%yout = column vector in R^3 at the end of step.

q=4;                       %order of RK method.

[m,n] = size(yin);
u1 = zeros(m,n);           %a(1,1)=0 for explicit rk method.
ytemp = yin + u1;          %lie group action is addition in the classical case.
```

```
k1 = lorenz(ytemp, sigma, rho, beta);
K1 = dexpinv(u1, k1, q);

u2 = 1/2*h*K1;
ytemp = yin + u2;
k2 = lorenz(ytemp, sigma, rho, beta);
K2 = dexpinv(u2, k2, q);

u3 = 1/2*h*K2;
ytemp = yin + u3;
k3 = lorenz(ytemp, sigma, rho, beta);
K3 = dexpinv(u3, k3, q);

u4 = h*K3;
ytemp = yin + u4;
k4 = lorenz(ytemp, sigma, rho, beta);
K4 = dexpinv(u4, k4, q);

v = h/6*(K1 + 2*K2 + 2*K3 + K4);                %approximation v ~ u(t_n+1)
yout = yin + v;                                  %y_n+1
tout = tin + h;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

The above function file implements two other function files `adj.m` and `dexpinv.m`. They are given below.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function w = adj(u,v,k)
%adj.m
%function file for matrix commutator w=ad^k_u(v).
%in the classical case, the algebra R^3 is commutative, adj^k_u(v)=0 for k>=1.
%k is the index for the adjoint operator ad^k.
if k==0
   w = v;
end
if k~=0
   w = 0;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function d = dexpinv(u, v, q)
%dexpinv.m
%dexpinv = sum_{k=0 to q-2}B_k/k!*ad^k_u(v)
%q is the order of the RK method used in RKMK.
%the smallest value of q is 2.

Ber = [1 -1/2 1/6 0 -1/30 0 1/42 0 -1/30 0 5/66];          %Bernoulli numbers.
```

```
if q==2
   dsum = v;
end
if q==3
   dsum = v - 1/2.*adj(u,v,1);
end
if (q==4 | q==5)
   dsum = v - 1/2.*adj(u,v,1) + 1/12.*adj(u,v,2);
end
d = dsum;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

To implement the integrator `exprkmk4_exp.m` to solve the Lorenz Equations, we implement it in a separate script file, displayed below.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all; close all; format long; clc;

global sigma rho beta;                    %parameters for the Lorenz Equations.
sigma=10; beta=8/3; rho=28;

t0 = 0;                                   %initial values.
y0 = [17;-21;54];                         %initial y(0).
tf = 10;                                  %final value.
h = 1/20;                                 %fixed stepsize.
tspan = [t0:h:tf]';                       %vector of independent t values.

tout3 = tspan;                            %Starting RKMK method.
yout3 = y0';
numstep=(tf-t0)/h;
t = tspan(1);                             %initialize variables.
y = y0;
for i = 1:numstep                         %solve the Lorenz Equations.
    [toutput, youtput] = exprkmk4_exp(t, y, sigma, rho, beta, h);
    yout3(i+1,:) = youtput';
    t = tspan(i+1);                       %update results.
    y = youtput;
end
clear numstep t y i toutput youtput;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

B.3 Special Orthogonal Matrix Differential Equations

From Section 4.3.1, Matlab's build-in function file **expm1.m** is displayed below, as found in the Matlab public directory, curtsy of Dr. C. Moler. More detail about the following Matlab function can be found in an updated version of a famous paper written by Moler and Van Loan in 1978, concerning the calculation of matrix exponentials [25].

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function E = expm1(A)
%EXPM1  Matrix exponential via Pade approximation.
%   E = EXPM1(A) is an M-file implementation of the built-in
%   algorithm used by MATLAB for the matrix exponential.
%   See Golub and Van Loan, Matrix Computations, Algorithm 11.3-1.
%
%   See also EXPM, EXPM2, EXPM3.

%   Copyright (c) 1984-98 by The MathWorks, Inc.
%   $Revision: 5.3 $  $Date: 1997/11/21 23:38:25 $

% Scale A by power of 2 so that its norm is < 1/2 .
[f,e] = log2(norm(A,'inf'));
s = max(0,e+1);
A = A/2^s;

% Pade approximation for exp(A)
X = A;
c = 1/2;
E = eye(size(A)) + c*A;
D = eye(size(A)) - c*A;
q = 6;
p = 1;
for k = 2:q
   c = c * (q-k+1) / (k*(2*q-k+1));
   X = A*X;
   cX = c*X;
   E = E + cX;
   if p
     D = D + cX;
   else
     D = D - cX;
   end
   p = ~p;
end
E = D\E;

% Undo scaling by repeated squaring
for k=1:s, E = E*E; end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

The two integrators RK4 `exprk4.m` and RK4 modified with compensated error addition `exprk4comp.m` are displayed below. They were used to solve Eq (4.5) in Section 4.3, where `Func.m` contains Eq (4.5).

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function [yout] = exprk4(xin, yin, h)
%exprk4.m
%function file for one step of an explicit RK4 method.
%Func.m contains DE we want to solve.
%yin = y value at the beginning of step.
%h = stepsize
%yout = y value at the end of the step.

k1 = Func(xin, yin);                                    %internal stage values.
k2 = Func(xin+1/2*h, yin+1/2*h.*k1);
k3 = Func(xin+1/2*h, yin+1/2*h.*k2);
k4 = Func(xin+h, yin+h.*k3);

yout = yin + h/6.*(k1 + 2.*k2 + 2.*k3 + k4);           %output values.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [ynew, z] = exprk4comp(xin, yin, h, zin)
%exprk4comp.m
%function file for one step of an explicit rk4 method,
%Func.m contains DE we want to solve.
%using compensated error addition.
%yin = y value at the beginning of step.
%h = stepsize
%zin = rounding error estimate from prevous step.
%ynew = y value at the end of the step.
%z = rounding error esimate from current step.

k1 = Func(xin, yin);                                   %internal stage values.
k2 = Func(xin+1/2*h, yin+1/2*h.*k1);
k3 = Func(xin+1/2*h, yin+1/2*h.*k2);
k4 = Func(xin+h, yin+h.*k3);

term = h/6.*(k1 + 2.*k2 + 2.*k3 + k4) + zin;          %term=delta
ynew = yin + term;
z = term - (ynew-yin);                                 %find round-off error z.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

The two integrators, RKMK4 and RKMK4 with compensated error addition are displayed below. They were used to solve Eq (4.5). Note that in this example, `func.m` is given by the function $f : \mathrm{SO}(3) \to \mathfrak{so}(3)$ in Section 4.3.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [yout] = exprkmk4(xin, yin ,h)
%exprkmk4.m
%function file for one step of explicit RKMK method of order 4.
%func contains f:SO(3)->so(3).
%manifold=G=SO(3) is a matrix Lie group.
```

```
%Lie algebra action, lambda(v,p)=expm(v)*p in G.
%xin = input indepedent value at beginning of step.
%yin = y value at the beginning of step.
%h = stepsize.
%yout = y value at the end of step.

q = 4;                                       %order of RK method.

[m,n] = size(yin);
u1 = zeros(m,n);                             %a(1,1)=0 for explicit rk method.
k1 = func(xin, expm(u1)*yin);                %lie group action = expm(u)*y
K1 = dexpinv(u1, k1, q);

u2 = 1/2*h.*K1;
k2 = func(xin+1/2*h, expm(u2)*yin);
K2 = dexpinv(u2, k2, q);

u3 = 1/2*h.*K2;
k3 = func(xin+1/2*h, expm(u3)*yin);
K3 = dexpinv(u3, k3, q);

u4 = h.*K3;
k4 = func(xin+h, expm(u4)*yin);
K4 = dexpinv(u4, k4, q);

v = h/6.*(K1 + 2.*K2 + 2.*K3 + K4);          %approximation v ~ u(t_n+1)
yout = expm(v)*yin;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [newy, z] = exprkmk4comp(xin, yin ,h, zin)
%exprkmk4comp.m
%function file for one step of exmplicit rkmk of order 4,
%with compensated error addition.
%func.m contains f:SO(3)->so(3).
%manifold=G=SO(3) is a matrix Lie group.
%Lie algebra action, lambda(v,p)=expm(v)*p in G.
%xin = input indepedent value at beginning of step.
%yin = y value at the beginning of step.
%h = stepsize.
%zin = rounding error estimate from previous step.
%newy = y value at the end of step.
%z = rounding error estimate from current step.

q = 4;                                        %order of RK method.

[m,n] = size(yin);
u1 = zeros(m,n);                              %a(1,1)=0 for explicit rk method.
k1 = func(xin, expm(u1)*yin);                 %lie group action = expm(u)*y
```

```
K1 = dexpinv(u1, k1, q);

u2 = 1/2*h.*K1;
k2 = func(xin+1/2*h, expm(u2)*yin);
K2 = dexpinv(u2, k2, q);

u3 = 1/2*h.*K2;
k3 = func(xin+1/2*h, expm(u3)*yin);
K3 = dexpinv(u3, k3, q);

u4 = h.*K3;
k4 = func(xin+h, expm(u4)*yin);
K4 = dexpinv(u4, k4, q);

%using 'first attempt' which failed to improve the accuracy of solutions.
%v = h/6*(K1 + 2*K2 + 2*K3 + K4);
%delta = (expm(v)-eye(m,n))*yin + zin;
%newy = yin + delta;
%z = delta - (newy - yin);

%With successful compensated error addition technqiue.
v = h/6*(K1 + 2*K2 + 2*K3 + K4);
delta = Ediff2(v)*yin + zin;
newy = yin + delta;
z = delta - (newy-yin);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Note that the RKMK integrators implement several separate files, including `adj.m`, `dexpinv.m`, and the compensated error file `Ediff2.m`. All 3 files are displayed below.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function w = adj(u,v,k)
%adj.m
%function file for matrix commutator w=ad^k_u(v).
%u and v are matrices
%k is the index for the adjoint operator ad^k.
if k==0
   w = v;
end
if k==1
   w = u*v-v*u;
end
if k==2
   w = u*u*v-2.*u*v*u + v*u*u;
end
if k==3
   w = u*u*u*v - 3.*u*u*v*u + 3.*u*v*u*u - v*u*u*u;
end
```

```
if k==4
   w = u*u*u*u*v - 4.*u*u*u*v*u + 6.*u*u*v*u*u - ...
       4.*u*v*u*u*u + v*u*u*u*u;
end
if k>4
   temp = u*u*u*u*v - 4.*u*u*u*v*u + 6.*u*u*v*u*u - ...
          4.*u*v*u*u*u + v*u*u*u*u;
   for i=5:k
       temp = u*temp-temp*u;
   end
   w = temp;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function d = dexpinv(u, v, q)
%dexpinv.m (inverse of dexp)
%dexpinv = sum_{k=0 to q-2}B_k/k!*ad^k_u(v)
%u and v are matrices.
%q is the order of the RK method used in RKMK.
%the smallest value of q is 2.

Ber = [1 -1/2 1/6 0 -1/30 0 1/42 0 -1/30 0 5/66];     %Bernoulli numbers.
if q==2
   dsum = v;
end
if q==3
   dsum = v - 1/2.*adj(u,v,1);
end
if (q==4 | q==5)
   dsum = v - 1/2.*adj(u,v,1) + 1/12.*adj(u,v,2);
end
d = dsum;                                             %output d.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function E = Ediff2(v)
%Ediff2.m
%function file for calculating expm(v)-Id,
%to find the round-off error estimate in RKMK4.

%v is a matrix.
%expm(v) is approximated by the Pade Approximation R(6,6),
%with scaling and squaring so that [expm(v/2^j)]^2^j.
[m,n] = size(v);

%Scale v by power of 2 so that its norm is < 1/2.
[f,e] = log2(norm(v, 'inf'));
s = max(0,e+1);
```

```
v = v/2^s;                                          %scaling v by 2^s.

v2 =  v*v; v3 = v2*v; v4 = v3*v; v5 = v4*v; v6 = v5*v;

if s==0                                             %no scaling of v.
   den = eye(m,n) - 1/2*v + 5/44*v2 - 1/66*v3 + 1/792*v4 - 1/15840*v5 +...
         1/665280*v6;
   w = den \ (v + 1/33*v3 + 1/7920*v5);             %pade expansion for exp-Id.
end
if s ~= 0
   den =  eye(m,n) - 1/2*v + 5/44*v2 - 1/66*v3 + 1/792*v4 - 1/15840*v5 +...
          1/665280*v6;
   w = den \ (v + 1/33*v3 + 1/7920*v5);             %pade expansion for exp-Id.

   for i = 1:s
       w = w*w + 2*w;
   end
end
E = w;                                              %output E.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

All of the above files are implemented by the following script file, to integrate the differential equation in Eq (4.5).

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all; format long; close all; clc;
t0 = 0;                                        %initial values.
y0 = [-0.30629995334276 -0.46674863567359 -0.82965417474937; ...
        -0.06581027265024 -0.85907704683087 0.50759790742481; ...
        -0.94965748909590 0.21007698280677 0.23241840438933];
tf = 10;                                       %final values.
h = 1/10;                                      %fixed stepsize.
tspan = [t0:h:tf]';                            %vector of independent t values.

%implement explicit rk4 -------------------------------------------------------
tout1 = tspan;                              %storage allocation.
yout1 = [];                                 %matrix stored as column vector.
numstep=(tf-t0)/h;
t = tspan(1);
y = y0;
yout1 = y0(:);                              %1st column of matrix is y0 vector.
for i = 1:numstep
   [youtput] = exprk4(t, y, h);             %implement exprk4.m
   yout1(:,i+1) = youtput(:);               %update values.
   t = tout1(i+1);
   y = youtput;
end
clear numstep t y i youtput;
```

```
%implement explicit rk4 with compensated error addition -----------------------
tout1comp = tspan;                                  %storage allocation.
yout1comp = [];                                     %matrix stored as column vector.
numstep=(tf-t0)/h;
t = tspan(1);
y = y0;
yout1comp = y0(:);                                  %1st column of matrix is y0 vector.
z = zeros(3,3);                                     %round-off error
for i = 1:numstep
   [youtput, znew] = exprk4comp(t, y, h, z);       %implement exprk4comp.m
   yout1comp(:,i+1) = youtput(:);                  %update values.
   t = tout1comp(i+1);
   y = youtput;
   z = znew;
end
clear numstep t y z i youtput;

%implementing explicit rkmk4 ---------------------------------------------------
tout2 = tspan;                                      %storage allocation.
yout2 = [];                                         %matrix stored as column vector.
numstep=(tf-t0)/h;
t = tspan(1);
y = y0;
yout2 = y0(:);                                      %1st column of matrix is y0 vector.
for i = 1:numstep
   [youtput] = exprkmk4(t, y, h);                   %implement exprkmk4.m
   yout2(:,i+1) = youtput(:);                       %update values.
   t = tout2(i+1);
   y = youtput;
end
clear numstep t y i youtput;

%implementing explicit rkmk4 with compensated error addition ------------------
tout2comp = tspan;                                  %storage allocation.
yout2comp = [];                                     %matrix stored as column vector.

numstep=(tf-t0)/h;
t = tspan(1);
y = y0;
yout2comp = y0(:);                                  %1st column of matrix is y0 vector.
z = zeros(3,3);                                     %round-off error
for i = 1:numstep
   [youtput, znew] = exprkmk4comp(t, y, h, z);     %implement exprkmk4comp.m
   yout2comp(:,i+1) = youtput(:);                  %update values.
   t = tout2comp(i+1);
   y = youtput;
   z = znew;
end
```

```
clear numstep t y z i youtput;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

B.4 Rigid Body Equations

The codes for the integrators written here for solving the rigid body problem are very similar to the codes which can be found in Section B.3 of this Appendix B. Hence, I shall only include the code for the RKMK method with Cayley mapping, as an example of how to construct RKMK integrators with other canonical coordinates, to solve the rigid body problem. Below is the integrator `exprkmk4_cay.m`.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [xout, Qout] = exprkmk4_cay(xin, Qin, Ls, h)
%exprkmk4_cay.m
%function file for one step of an explicit rkmk4 method,
%on the generic DE of the rigid body.
%Q'=f(Q)*Q, where Q in SO(3), f(Q)=QI^{-1}Q^{-1}Ls,
%Ls=angular momentum in space coordinates=constant vector.
%Let's take Ls=s0, the initial angular momentum in the body coordinate.
%Coordinate map used here is the cayley coordinate of the first kind.
%xin = input independent variable.
%Ls = column vector of Ls=s0 at beginning of integration.
%Qin = input SO(3) matrix, Q(0)=Id.
%h = stepsize.
%xout = x value at the end of step.
%Qout = SO(3) matrix at the end of step.

%integrate Q'=f(Q)*Q. action=cay(v)*p.
%inertia tensor for the rigid rectangular block.
global I1 I2 I3; I = diag([I1 I2 I3],0);

[m,n] = size(Qin);
u1 = zeros(m,n);                           %a(1,1)=0 for explicit rk method.
Qtemp = cay(u1)*Qin;                       %lie group action=cay(u)*Q.
k1 = Rw(Qtemp*inv(I)*inv(Qtemp)*Ls);
K1 = dcayinv(u1, k1);

u2 = 1/2*h*K1;
Qtemp = cay(u2)*Qin;
k2 = Rw(Qtemp*inv(I)*inv(Qtemp)*Ls);
K2 = dcayinv(u2, k2);

u3 = 1/2*h*K2;
Qtemp = cay(u3)*Qin;
k3 = Rw(Qtemp*inv(I)*inv(Qtemp)*Ls);
K3 = dcayinv(u3, k3);
```

```
u4 = h*K3;
Qtemp = cay(u4)*Qin;
k4 = Rw(Qtemp*inv(I)*inv(Qtemp)*Ls);
K4 = dcayinv(u4, k4);

v = h/6*(K1 + 2*K2 + 2*K3 + K4);          %approximation v ~ u(t_n+1)

Qout = cay(v)*Qin;                         %output Q_n+1.
xout = xin + h;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

The above function file `exprkmk4_cay.m` implements 3 other function files, `Rw.m`, `dcayinv.m`, and `cay.m`. The last two files are needed due to the change in canonical coordinates from the exponential map to the Cayley map, while `Rw.m` is a skew-symmetric matrix in $\mathfrak{g} = \mathfrak{so}(3)$ which acts as the map $f : \mathcal{M} \to \mathfrak{g}$, denoted with an overline in the rigid body discussion. This is used by the RKMK method to solve the differential equation in Eq (4.42).

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function U = cay(u)
%cay.m
%The Cayley coordinates of the first kind is a mapping from g-> G.
%It is the first diagonal Pade approximant to the exponential,
%used to replace the matrix exponential in the classical RKMK method.
%u = matrix input u.
%U = matrix output u.

U = inv(eye(3,3)-u./2) * (eye(3,3)+u./2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function C = dcayinv(u, v)
%dcayinv.m
%The differential of the Cayley mapping.
%dcayinv:g -> g,
%used to replace dexpinv in the original RKMK method.
%u = input u matrix.
%v = input v matrix.
%C = output dcayinv matrix.

C = v - 1/2.*adj(u,v,1) - 1/4.*u*v*u;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [R] = Rw(w)
%Rw.m
%function file for writing the angular velocity vector as a skew-symmetric
%matrix, used in the rigid body problem.
%w = column vector of angular velocity.
```

```
%R = 3x3 skew symmetric matrix R[w] in so(3).

R = [0 -w(3) w(2);
     w(3) 0 -w(1);
     -w(2) w(1) 0];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Note that to find the angular velocity $\omega^b(t)$ and the position $r^s(t)$ of the rigid body, we need to use the follow script file. In this case, it is integrating the generic presentation of the Euler equations of motions, given in Eq (4.42), using the RKMK method with the Cayley mapping. Note that the rigid body is a rectangular block defined in Section 4.5.5.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all; close all; format long; clc;

global I1 I2 I3                %moments of inertias for the rectangular block.
I1 = 52; I2 = 40; I3 = 20;

t0 = 0;                        %initial time.
y0 = [5 0 0 -1 -1 1 1 1 -1 -1 1;%inital position.
    0 5 0 2 -2 -2 2 2 2 -2 -2;
    0 0 5 3 3 3 3 -3 -3 -3 -3];
w0 = [0.02; 5; 0.1];           %initial angular velocity in body coordinates.
tf = 40;                       %final time.
h = 1/40;                      %fixed stepsize.
tspan = [t0:h:tf]';            %vector of independent t values.

%initial angular momentum, s=Iw, in body coordinates.
%used as the initial angular momentum in space coordinates at t=0, Ls.
s0 = [I1*w0(1); I2*w0(2); I3*w0(3)];

%implementing RKMK method with Cayley mapping ---------------------------------
sout2b = s0;                   %angular momentum, body coordinates.
wout2b = w0;                   %angular velocity, body coordinates.
yout2b = y0(:);                %position, space coordinates.
tout2b = tspan;                %time.
numstep = (tf-t0)/h;
t = tspan(1);
Ls = s0;      %initial L in body coordinates=initial L in space coordinates.
Q = eye(3,3);                  %Q(0)=Id.
I = diag([I1 I2 I3],0);        %inertia tensor.
for i = 1:numstep
    [toutput, Qoutput] = exprkmk4_cay(t, Q, Ls, h);
    sout2b(:,i+1) = inv(Qoutput)*Ls;                 %body coordinate.
    wout2b(:,i+1) = inv(I)*sout2b(:,i+1);
    ytemp = Qoutput * y0;
    yout2b(:,i+1) = ytemp(:);
    t = tspan(i+1);
```

```
    Q = Qoutput;
end
clear numstep t Q I toutput Qoutput ytemp;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Bibliography

[1] R. Abraham, J.E. Marsden, T. Ratiu. *Manifolds, Tensor Analysis, And Applications*, second edition, Applied Mathematical Sciences 75, Springer-Verlag, New York (1988).

[2] T. Apostol. *Linear Algebra : A First Course, With Applications To Differential Equations*, Wiley-Interscience Publication, John Wiley & Sons Ltd, New York (1997).

[3] G.G.A. Bäuerle, E.A. de Kerf. *Lie Algebras - Part 1 Finite And Infinite Dimensional Lie Algebras And Applications In Physics*, Studies in Mathematical Physics 1, North-Holland, Amsterdam (1990).

[4] P. Blanchard, R. Devaney, G.R. Hall. *Differential Equations*, first edition, Brooks/Cole Publishing Company, California (1998).

[5] R.L. Burden, J.D. Faires. *Numerical Analysis*, fifth edition, PWS Publishing Company, Boston (1993).

[6] J.C. Butcher. *Numerical Analysis Of Ordinary Differential Equations : Runge-Kutta And General Linear Methods*, Wiley-Interscience Publication, John Wiley & Sons Ltd, Chichester (1987).

[7] J.C. Butcher. *Numerical Methods For Ordinary Differential Equations*, to be published.

[8] E. Celledoni, B. Owren. Lie Group Methods For Rigid Body Dynamics And Time Integration On Manifolds, to be published in *Comput. Methods Appl. Mech. Engrg.*

[9] Y. Choquet-Bruhat, C. DeWitt-Morette, M. Dillard-Bleick. *Analysis, Manifolds And Physics*, second edition, North-Holland Publishing Co., Amsterdam (1982).

[10] J.R. Dormand, P.J. Prince. A Family Of Embedded Runge-Kutta Formulae, *J. Comput. Appl. Math.* **6** (1980), 19–26.

[11] K. Engø. On The Construction Of Geometric Integrators In The RKMK Class, *BIT* **40** (2000), 41–61.

[12] K. Engø, A. Marthinsen, H.Z. Munthe-Kaas. *DiffMan User's Guide Version 2.0*, Department of Informatics, University of Bergen, Norway (2000).

[13] E. Hairer. Geometric Integration Of Ordinary Differential Equations On Manifolds, *BIT* **41** (2001), 996–1007.

[14] E. Hairer, C. Lubich, G. Wanner. *Geometric Numerical Integration, Structure-Preserving Algorithms For Ordinary Differential Equations*, Springer Series in Computational Mathematics 31, Springer-Verlag, Berlin (2002).

[15] A. Iserles. Insights, Not Just Numbers, *Proceedings of the 15th IMACS World Congress vol. II*, Wissenschaft & Technik Verlag, Berlin (1997), 1–9.

[16] A. Iserles. Numerical Analysis In Lie Groups, in *Foundations of Computational Mathematics (Oxford, 1999)*, Cambridge University Press, Cambridge (2001), 105–123.

[17] A. Iserles. On Cayley-transform Methods For The Discretization Of Lie-group Equations, *Found. Comput. Math.* **1** (2001), 129–160.

[18] A. Iserles. Expansions That Grow On Trees, *Notices Amer. Math. Soc.* **49** (2002), 430–440.

[19] A. Iserles, H. Munthe-Kaas, S. Nørsett, A. Zanna. Lie-group Methods, *Acta Numerica* **9** (2000), 215–365.

[20] A. Iserles, A. Marthinsen, S.P. Nørsett. On The Implementation Of The Method Of Magnus Series For Linear Differential Equations, *BIT* **39** (1999), 281–304.

[21] A. Iserles, S.P. Nørsett, A.F. Rasmussen. Time-symmetry And High-order Magnus Methods, *Appl. Numer. Math.* **39** (2001), 379–401.

[22] G.H. Golub, C.F. Van Loan. *Matrix Computations*, second edition, Johns Hopkins Series in the Mathematical Sciences 3, Johns Hopkins University Press, Baltimore (1989).

[23] S. Lang. *Fundamentals Of Differential Geometry*, Graduate Texts in Mathematics 191, Springer-Verlag, New York (1999).

[24] J.E. Marsden, T.S.Ratiu. *Introduction To Mechanics And Symmetry. A Basic Exposition Of Classical Mechanical Systems*, first edition, Texts in Applied Mathematics 17, Springer-Verlag, New York (1994).

[25] C. Moler, C. Van Loan. Nineteen Dubious Ways To Compute The Exponential Of A Matrix, Twenty-Five Years Later. To be re-published in *SIAM Review* 2003. (Original paper appeared in *SIAM Review* **20** (1978), 801-836).

[26] H. Munthe-Kaas. Lie-Butcher Theory For Runge-Kutta Methods, *BIT* **35** (1995), 572–587.

[27] H. Munthe-Kaas. Runge-Kutta Methods On Lie Groups, *BIT* **38** (1998), 92–111.

[28] H. Munthe-Kaas. High Order Runge-Kutta Methods on Manifolds, *Appl. Numer. Math.* **29** (1999), 115–127.

[29] H. Munthe-Kaas, A. Zanna. Numerical Integration of Differential Equations On Homogeneous Manifolds, in *Foundations of Computational Mathematics (Rio de Janeiro, 1997)*, 305–315, Springer-Verlag, Berlin (1997).

[30] P.J. Olver. *Applications Of Lie Groups To Differential Equations*, second edition, Graduate Texts in Mathematics 107, Springer-Verlag, New York (1993).

[31] B. Owren, A. Marthinsen. Integration Methods Based On Canonical Coordinates Of The Second Kind, *Numer. Math.* **87** (2001), 763–790.

[32] S.H. Strogatz. *Nonlinear Dynamics And Chaos : With Applications To Physics, Biology, Chemistry, And Engineering*, Perseus Books, Massachusetts (1994).

[33] S.M. Tan. *Lecture Notes from Physics 310 Classical Mechanics*, Department of Physics, The University of Auckland.

[34] V.S. Varadarajan. *Lie Groups, Lie Algebras, And Their Representations*, Prentice-Hall Series in Modern Analysis, Prentice-Hall, New Jersey (1974).